MIT-T-87-003 C2

N.M. Patrikalakis
P.V. Prakash

Computation of Algebraic Polynomial Parametric Surface Intersections

LOAN COPY ONLY

CIRCULATING COPY
Sea Grant Depository

MIT Sea Grant
College Program

Massachusetts
Institute of Technology
Cambridge, MA 02139

MITSG 87-19
December, 1987

COMPUTATION OF ALGEBRAIC AND POLYNOMIAL PARAMETRIC SURFACE INTERSECTIONS

BY

N.M. PATRIKALAKIS

P.V. PRAKASH

$6.00

MIT Sea Grant
College Program

Massachusetts Institute
of Technology
77 Massachusetts Ave.
Cambridge, MA 02139

MITSG 87-19
December, 1987
NA86AA-D-SG089
R/T 16

Table of Contents

List of Figures

List of Tables

ABSTRACT

In this work we present the basic elements of a new algorithm to trace a planar algebraic curve f(u,v) = 0 within a rectangular parallelogram arising in the context of automatic interrogation of intersections of algebraic surfaces and piecewise continuous rational polynomial parametric surface patches. The method combines the advantageous features of analytic representation of the governing equation in the Bernstein basis with adaptive subdivision techniques and the a priori computation of turning and singular points to provide the basis for a reliable and efficient solution procedure. Representative results from the application of the method in tracing well known algebraic curves and in computing intersections of typical algebraic and rational polynomial surfaces are also presented.

ACKNOWLEDGMENTS

This work was supported in part by the MIT Sea Grant College Program, the General Electric Company and the National Science Foundation.

AUTHORS

N.M. Patrikalakis is an Assistant Professor of Ocean Engineering at MIT, the Doherty Professor in Ocean Utilization, 1988-1990, and a member of the Design Lab.

P.V. Prakash is a Doctoral Student in MIT's Department of Ocean Engineering.

1. INTRODUCTION AND OUTLINE

The ability to discover and describe all features of an unknown implicit polynomial (algebraic) curve F(u,v) = 0 within a finite domain is a high priority issue in computer aided engineering. Contouring and sectioning, general polynomial surface intersections and silhouette evaluations can all, in principle, be cast in the form of the above fundamental problem. Such computer programs are essential tools for the representation and interrogation of the external and internal shape of complex engineering objects during different phases of design, analysis and fabrication. The reliable and efficient interrogation of an algebraic curve is, therefore, one of the key elements in the development of reliable modern modeling and manufacturing systems in a unified computer environment [51, 24, 12].

The representation and interrogation of algebraic curves of arbitrary degree is a major stepping stone in extending the coverage of solid modelers to a richer family of primitives beyond those available today. In the solid modeling context, algebraic curves arise in the representation and processing of trimmed surface patches [17, 19, 11] and in the construction and verification of boundary representations of solids. They also arise in analysis, such as in the derivation of finite element discretizations and in fabrication, such as in control of cutting and welding robots [12].

This report presents the basic elements of a new algorithm allowing automatic interrogation of planar algebraic curves within a rectangular parallelogram, arising in the context of intersections of algebraic surfaces and piecewise continuous rational polynomial parametric surface patches such as non-uniform rational parametric B-spline patches. Our method exploits the advantageous features of analytic representation of the governing equation in the Bernstein basis with adaptive subdivision and the a priori computation of turning and singular points of the curve and provides the basis for a reliable and efficient solution procedure. In this report, we focus primarily on robustness issues of the proposed algorithm. In particular, we address the problem of reliable derivation of the correct connectivity of the curve in the presence of singularities and small isolated loops. Singularities and small loops are very common features of algebraic curves arising in a geometric modeling context and significantly affect the reliability and overall performance of current state-of-the-art algorithms.

This report is structured as follows

- Section 2 provides a brief review of the state-of-the-art in computing surface intersections highlighting some of the key contributions from different approaches to the problem.

- Section 3 provides a brief introduction to the major steps of our method and the motivation and rationale behind its development.
- Section 4 formulates the problem of computing intersections of algebraic surfaces and piecewise continuous rational polynomial surface patches and applies a novel method to represent algebraic curves in the Bernstein basis.
- Section 5 describes a method to compute the significant points of the curve, i.e. turning, singular and border points, and highlights some of the issues involved in the solution of univariate polynomial equations.
- Section 6 describes a method of tracing an algebraic curve represented in the Bernstein basis with known significant points by employing adaptive subdivision and faceting techniques.
- Section 7 presents examples from the interrogation of well known planar algebraic curves and of intersections of low order algebraics and typical rational polynomial surface patches expressed in the B-spline basis.
- Section 8 critiques the proposed method and provides recommendations for further research.

This report also includes the following Appendices

- Appendix I summarizes the formulas needed in transforming curves from the monomial to the Bernstein basis.
- Appendix II outlines a method to estimate the distance of a point from an algebraic surface.
- Appendix III discusses the classification of singular points of algebraic curves.
- Appendix IV provides a method to directly eliminate the common variable between two univariate polynomials expressed in the Bernstein basis.
- Appendix V outlines Kahan's summation technique allowing improved accuracy computation of large scale series summations.
- Appendix VI summarizes the Oslo algorithm for the non-uniform subdivision of B-spline curves and surfaces.
- Appendix VII summarizes a root finding technique for a univariate polynomial expressed in the Bernstein basis employing the variation diminishing property.

2. BRIEF LITERATURE REVIEW

In this Section, we briefly review some of the literature on surface-to-surface intersection algorithms primarily from the robustness point of view. The solution methods reviewed here can be classified in three main categories- analytic, lattice evaluation and marching and subdivision. In this report, we provide only a summary of the main ideas, advantageous features and outstanding problems in each of the different techniques.

2.1 Analytic Methods

Analytic methods rely on the derivation of a mathematically exact equation describing the intersection of two surfaces. For polynomial surfaces, the resulting equation in general is an implicit polynomial in two variables. This equation can, in principle, be obtained by elimination of one Cartesian coordinate for the case of two implicit surfaces [40] or by elimination of three Cartesian coordinates for the case of an implicit surface intersecting a parametric surface [18]. In the first case, an equation for the projection of the intersection curve on one coordinate plane is obtained and the inversion algorithm of [41], modified to resolve singularities, is needed to compute the third coordinate. A degeneracy occurs when the intersection curve is planar and the direction of projection is parallel to the plane of the curve in which case the projection is unbounded and needs to be restricted by considering its intersection with the corresponding silhouette of one of the surfaces. In the second case, the resulting intersection equation is expressed in the parameter space of one patch. When the implicit surfaces of interest are actually bounded, (i.e. they are algebraic patches [42, 38] or they resulted from implicitization of parametric patches [41]), points which are found to satisfy the intersection equation within the space of one patch should be tested to verify if they lie on the appropriate portion of the other surface. Some of the issues arising from the actual numerical evaluation of the intersection equation using the above methods are discussed in Section 4 of this report.

In practical situations, once the equation describing the intersection curve is obtained as above, it must be traced. For special cases, the resulting implicit equations can be solved in terms of explicit expressions involving radicals [34] or in terms of rational parametric polynomials for curves with genus zero [1]. Once the range of the independent variable in such cases is determined, the above explicit equations can be used to trace the intersection curve. However, for general cases explicit solutions in terms of elementary functions are impossible [41]. Algebraic curves with integer coefficients can be analysed using the cylindrical algebraic decomposition algorithm [3, 4, 5, 6] implemented in rational arithmetic thereby eliminating round-off error contamination of the solution. This method, although providing a guarantee that the solution is topologically reliable, is impractical, at present, because of its very large memory requirements and poor efficiency. In addition, algebraic curves with integer coefficients are, unfortunately, not general enough for geometric modeling, where simple rotation of intersecting primitives creates curves with possibly irrational or transcendental coefficients. The above considerations suggest the need to revert to one of the methods addressed in the sequel.

2.2 Lattice Evaluation and Marching Methods

Lattice evaluation methods reduce the dimensionality of surface-to-surface intersection problems by computing intersections of a number of parametric curves of one surface with the other surface followed by connection of the resulting discrete intersection points to form different solution branches [48]. For intersections of parametric patches, the method reduces to the solution of a large number of independent systems of three nonlinear equations in three unknowns. For intersections of parametric polynomial patches with algebraic surfaces it reduces to the computation of the real roots of a large number of independent polynomials within an interval. For reasons similar to those outlined in Sections 4 and 5, the numerical derivation of the governing equations to be solved is a key issue in the robustness of the method. Although the technique is partly parallelizable, the solution of each independent polynomial equation or system of equations is difficult. For univariate polynomials, no initial estimate of the solutions are required while for systems of nonlinear equations, available numerical techniques (such as Newton and minimization methods) require good initial approximations for convergence. This is an important disadvantage. Using elimination techniques [41], systems of nonlinear polynomial equations can be reduced to the solution of high degree univariate polynomials. The issues arising from such a process and with the reliable solution of high degree polynomials are discussed in Section 5. By definition of lattice evaluation methods, the reduction of the dimensionality of the problem involves an initial choice of grid resolution, which, in turn, may lead the method to miss important features of the solution, such as small loops and isolated points. Finally, the second element of the method involves connection of discrete solution points to form solution branches. This, typically, requires determining adjacency on the basis of minimum mutual distance which may lead to incorrect connectivity particularly near singular points. Derivative information may be employed to enhance the reliability of the method, but requires careful empirical tuning, which may be case dependent.

Marching methods involve generation of sequences of points of an intersection curve branch by stepping from a given solution point in a direction prescribed by the local differential geometry. Marching methods are relatively simple to implement and efficient as they do not require solution of large numbers of nonlinear equations. However, they require **starting points** for **every** branch and a stepping size which is case dependent and difficult to determine. Incorrect step size may lead to erroneous connectivity of solution branches or even to endless looping, as illustrated in [21]. Simple marching methods, not employing appropriate expansions of implicit polynomials near singular points, may also fail near such points. The desingularization method based on birational transformations outlined in [27, 7] provides an elegant rectification of this type of failure in marching methods.

Reliability of lattice evaluation and marching methods requires the determination of **all** significant points described in detail in Section 5 of this report, first introduced in [18] in a surface intersection context. The significant points include border, turning and singular points. The spacing of these points should be beneficial in the selection of, possibly, non-uniform **grid** size for lattice evaluation methods and **stepping size** and **initial points** for **every** branch in marching methods. Knowledge of significant points and the multiplicity of singular points also provides an independent count of the number of monotonic branches between significant points [18]. This count can confirm the number of branches obtained using lattice evaluation and marching methods and provides added confidence in the results of these methods. Such a verification is, practically, very important for the success of these methods because of their proclivity to erroneous topological connectivity. Encouraging results using such a method were reported in plane sectioning of low order parametric patches [18]. As pointed out in [18], the above method does not establish the proper number of branches in the presence of tacnodes, i.e. cusps with a tangent direction which is a multiple tangent of the curve; and it requires rotation of the curve to handle branches between singular points without intervening turning points, as it cannot start at singular points. Starting at singular points requires additional information which can be worked out using the desingularization procedure described in [27, 7]. The method to estimate the number of monotonic branches between significant points also requires a minor modification when turning and singular points are also border points. Concluding our review for lattice evaluation and marching methods, we should also point out the extreme importance of reliable computations of significant points, discussed in Section 5 of this report.

Barnhill et al [9] also use a marching method to trace the curve of intersection of two parametric surface patches which are not restricted to be rational polynomial providing the basis for a very general solution procedure. The method only requires a procedure to evaluate the surface position and its partial derivatives at any point and employs lattice evaluation, subdivision and Newton methods to determine starting points for different branches of the intersection to be traced by a marching method. As pointed out in [9], the method does not handle the intersection of partially coincident surfaces (degenerate case) and tangent surfaces (singularities and very small loops).

2.3 Subdivision Methods

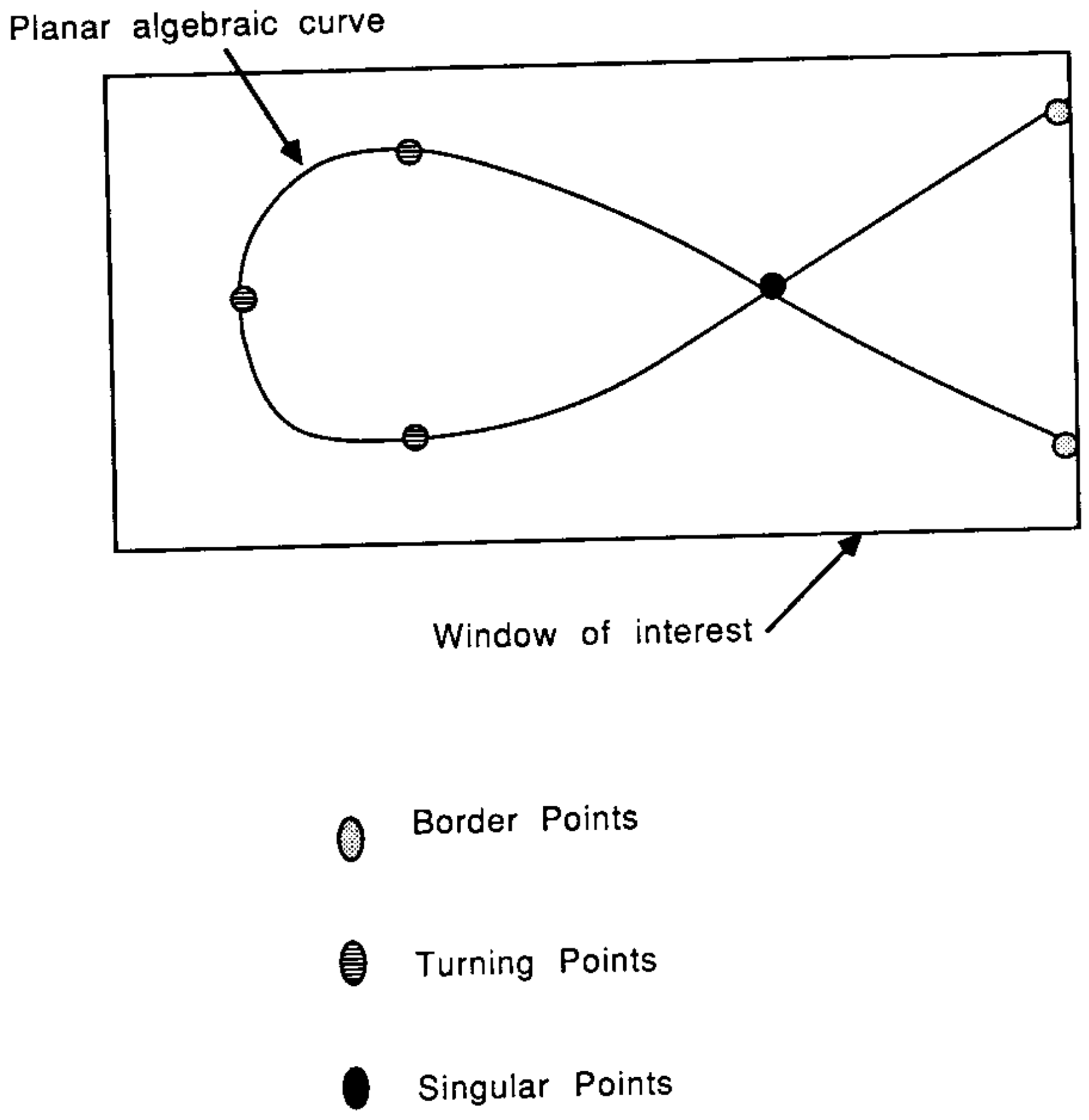
The main idea of subdivision methods involves recursive decomposition of the original intersection problem into simpler similar problems until a level of simplicity is reached, which allows simple direct solution, such as solution of linear algebraic equations (e.g. plane/ plane intersection). In solid modeling, this is followed by a connection phase of the individual solutions to form the complete solution. By definition, the application of the method requires that the problem be subdividable, and, at each stage, there must be some recognizable reduction in the complexity of the problem. Subdivision methods allow the computation of intersections of polynomial surfaces used in geometric design applications. Initially conceived in the context of intersections of polynomial parametric surfaces [31], they can be extended to the computation of algebraic/polynomial parametric and algebraic/algebraic surface intersections. This extension is a natural result of our reformulation of algebraic curves in terms of the Bernstein basis within a rectangular window and their interpretation as intersections of Bezier surfaces and a plane as explained in Sections 3 and 4 of this work [38]. Subdivision methods are also simple to implement and, most importantly, convergent in the limit, not suffering from many of the degeneracies of simple marching techniques and other, purely numerical, solution methods such as Newton's iterative procedures. Notably, subdivision techniques do not require starting points as marching methods, an important advantage from the reliability point of view. Many elements of subdivision techniques are also **parallelizable**, which is an important advantage for future large scale real time applications. A subdivision procedure to perform Boolean operations on objects bounded by B-spline surfaces has been studied in [46]. Finally, the non-uniform subdivision, studied in [13, 35], allows easy selective refinement of the solution providing the basis for an adaptive technique as described in Section 6 of this document. The major disadvantage of subdivision techniques is that, in actual implementations with finite subdivision steps, correct connectivity of solution branches near singular points is difficult to guarantee and extraneous small loops may be present in the approximation of the solution. In general, features of the solution smaller than the final subdivision size, are not resolved [18]. For example, in the method proposed in [21], the implicit intersection curve, expressed in the Bernstein form within a square, is subdivided using a quadtree approach until each cell contains **one monotonic** piece of the curve. The presence of such a simple piece of the curve within a cell can be detected by interrogation of the Bernstein coefficients of the curve for that cell. Simple curve segments can, in theory, be traced using marching. However, the simplicity criterion fails, for example, at self-intersection points and tacnodes and the algorithm recurses until the cell size is the limiting precision of the machine. This is an unsatisfactory feature of this method. A comprehensive review of different types of subdivision methods can be found in [18]. An important procedural variant of the subdivision- polyhedron faceting methods involves degree reduction and subdivision to reach the simplicity of plane/ plane intersections [39, 44]. The efficiency of degree reduction and other intersection techniques is discussed in [43].

The method proposed in the following sections of this report, coupling analytic description of the intersection curve expressed in the Bernstein basis, adaptive subdivision and faceting techniques with a priori determination of significant points alleviates many of the shortcomings of available solution procedures and has the potential to provide a more reliable method to compute surface-to-surface intersections.

3. BRIEF DESCRIPTION OF THE METHOD

In this report we consider the intersection of a rational parametric B-spline patch with an algebraic surface. The intersection between two such surfaces can be expressed as a set of implicit polynomial equations in the parameters of the patch. These implicit polynomial equations are normally expressed in the power basis and are restricted externally by the parametric range of polynomial spans of the B-spline patch and describe the three-dimensional curve of intersection as a planar algebraic curve in the parametric variables of the patch. The particular intersection procedure developed here transforms this representation to the Bernstein basis in order to exploit the geometric and computational properties of this representation. This is possible since only a piece of the curve represented by the implicit equation is actually involved in the intersection as expressed by the limited range of the parametric variables. The Bernstein representation of the curve allows this planar curve to be interpreted as the intersection of a parametric Bernstein-Bezier surface with a plane (hereafter called the control surface and control plane). Polyhedral faceting can then be used to construct an approximation to the image of the intersection curve in the parametric space. In particular, the use of the control polyhedron for this purpose is suitable as it converges to the control surface in the limit, in a very robust manner, and furthermore, possesses properties useful in the intersection problem. However, the presence of singularities in the original curve of intersection leads to major difficulties in tracing such curves in their neighborhood and in providing the true topology of the curve. Singularities also form the major common cause of failure of most well known intersection algorithms as discussed in Section 2. In this work, we deal only with curves having a finite number of singular points. The problem of handling singularities manifests itself in different ways for the methods discussed in that section. For polyhedral faceting methods, the complexity of the surface near singularities cannot, in general, be approximated by planar facets. The proposed method requires the a priori computation of turning and singular points in order to assist polyhedral faceting methods to provide the correct connectivity of the curve (see Figure 3-1). The border points which define the intersections of the curve with the boundaries of the rectangular window of interest are also determined to facilitate reliable connection between curves of adjacent patches. The control surface is subsequently split along parametric lines at the coordinates of singular, turning and border points. In this manner, the original control surface is transformed into a matrix of smaller control surfaces the intersections of which with the control plane have singular points only at their corner points and, hence, all curve complexity is isolated at the

Figure 3-1: A planar implicit curve



corners of the smaller surface patches. Turning points may either occur at corner points of the smaller control surfaces or they may cover one or more of their borders, if there is an infinity of such points. Each smaller control patch is then intersected with the control plane using a triangulation of the polyhedron to obtain one or more connected sequences of points in the domain of the patch. The points are tested for accuracy in three-dimensional space and based on the points failing an accuracy criterion, the polyhedron is adaptively refined using the Oslo subdivision algorithm. The resulting polyhedron is a closer approximation of the control surface and provides the means of progressively increasing the accuracy of the points on the intersection curve by iteration. The solutions from each one of these patches are subsequently combined into larger connected sequences of points which together form the solution of the intersection problem. Finally, the solution points in parametric space are mapped into the three-dimensional space. The solution concludes in the form of a linear spline approximation of the intersection curve in both three-dimensional space and in the parametric space of the patch.

4. FORMULATION OF THE GOVERNING EQUATIONS

4.1 Definitions

The intersection problem studied here is between piecewise continuous rational polynomial parametric patches and algebraic surfaces. In particular, we will consider intersections of non-uniform rational parametric B-spline surface patches with algebraic surfaces.

Let us consider a general Non-Uniform Rational B-Spline (NURBS) surface patch $\mathbf{Q}_{k,l}(u,v)$ of maximum degree k and l in the parametric variables u and v [22, 47]

$$\mathbf{Q}_{k,l}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{ij}N_{i,k+1}(u)\,N_{j,l+1}(v) \quad \textit{where} \quad u \in [u_a,u_b],\ v \in [v_a,v_b] \tag{1}$$

where $\mathbf{P}_{ij}$ are homogeneous coordinates of the (m+1) x (n+1) control points given in a world coordinate system and $m \geq k$, $n \geq l$. The three-dimensional coordinates of a point u,v of the patch are obtained by dividing each of the first three homogeneous coordinates of $\mathbf{Q}_{k,l}(u,v)$ by its fourth homogeneous coordinate. The fourth homogeneous coordinate of each control point $\mathbf{P}_{ij}$ is assumed to be positive. $N_{i,k}(u)$ and $N_{j,l}(v)$ are the B-spline basis functions defined over non-uniform knot vectors in the u and v directions given by $\{u_0,\dots u_{k+m+1}\}$ and $\{v_0,\dots v_{l+n+1}\}$ respectively. The B-spline basis functions may be computed by de Boor's recursion [15]

$$N_{i,p}(t) = \frac{t-t_i}{t_{i+p-1}-t_i}N_{i,p-1}(t) + \frac{t_{i+p}-t}{t_{i+p}-t_{i+1}}N_{i+1,p-1}(t)$$

and

$$N_{i,1}(t) = \begin{cases} 1 & \textit{if } t \in [t_i,t_{i+1}) \\ 0 & \textit{otherwise} \end{cases} \tag{2}$$

The algebraic surface, G, is expressed as an implicit polynomial equation in the homogeneous coordinates x,y,z & w

$$G(x,y,z,w) = \sum_{i=0}^{q}\sum_{j=0}^{q-i}\sum_{k=0}^{q-i-j} D_{ijk}x^i y^j z^k w^{q-i-j-k} = 0 \tag{3}$$

where x/w, y/w, z/w are the three-dimensional Cartesian coordinates in the world coordinate system and the degree of the surface is q. An algebraic surface of degree q involves up to d = (q+1)(q+2)(q+3)/6 coefficients and up to d - 1 degrees of freedom, because one of the coefficients in (3) can be chosen arbitrarily without modifying the algebraic surface. In the sequel, it is convenient to assume that the coefficients D_{ijk} of algebraic surfaces are normalized using the maximum $|D_{ijk}|$ so that $|D_{ijk}| \leq 1$.

4.2 Preliminary Remarks

The intersection of the above two surfaces can be defined as the set of points of patch (1) which lie on the surface represented by (3). Hence the general point on (1) is substituted into (3) to give (4), the general equation of the intersection of the two surfaces

$$F(u, v) = 0 \tag{4}$$

Such an equation, in general, represents an algebraic curve in the variables u,v, where u,v are restricted to the parametric space of patch (1). The maximum degree in u is kq and in v is lq. The substitution of (1) in (3), while conceptually simple is very difficult to perform in practice for anything but the simplest algebraic surface, ie. the plane. For degree two algebraic surfaces, the quadrics, the substitution involves the squaring of piecewise polynomial functions and is unattractive for use. For higher degree algebraic surfaces, it involves raising piecewise polynomials to higher degrees. Further, the general form of (3) involves products of x, y, z and w making the substitution very complex. In principle, any reduction in the complexity of (3) by suitable transformations, such as translation and rotation, is useful for implementation. Normally, classical algebraic surfaces in a solid modeling environment are specified in a local (natural) coordinate system which is in turn related to the world coordinate system by translation and rotation and in which the algebraic surface equation assumes a compact form [45].

These considerations suggest that it is convenient to perform the following transformations

1. Transformation of the parametric patch equation to the local (natural) system of the algebraic surface.
2. Conversion of the B-spline surface patch into its component polynomial patches.

When the algebraic surface is a plane, such transformations are not needed as the parametric patch equation can be directly substituted in the implicit plane equation. The above two transformations are described in Sections 4.2.1 and 4.2.2.

4.2.1 Transformation 1

Let $r_0 = [r_{0x}\ r_{0y}\ r_{0z}\ 1]$ be a vector specifying the location of the origin and $(\mathbf{u}_x, \mathbf{u}_y, \mathbf{u}_z)$ be three unit vectors specifying the orientation of the local (natural) coordinate system of the algebraic surface in the world coordinate system. The algebraic surface (3) can now be specified in the local system by a simpler equation

$$H(x,y,z,w) = \sum_{i=0}^{q}\sum_{j=0}^{q-i}\sum_{k=0}^{q-i-j} C_{ijk}x^i y^j z^k w^{q-i-j-k} = 0 \tag{5}$$

where the C_{ijk} are assumed to be normalized using the maximum $|C_{ijk}|$ so that $|C_{ijk}| \leq 1$. It is easy to define the local (natural) coordinate system for the commonly used algebraic surfaces in solid modeling such as planes, spheres, circular cylinders and cones and torii (also called natural surfaces). For the plane, it is any orthogonal system having two axes on the plane. For the sphere, any orthogonal coordinate system with origin located at the center of the sphere is a natural coordinate system. For the circular cylinder, it is any orthogonal system that has one axis aligned with the central axis of rotational symmetry. For the circular cone it is any orthogonal system that has one axis aligned with the central axis of rotational symmetry with the origin at the vertex of the cone. For the torus, it is any orthogonal system with origin at the centroid of the torus and one axis aligned with the axis of rotational symmetry. The 4x4 matrix, M_{aw}, which transforms coordinates in the local (natural) coordinate system of the algebraic surface to the world coordinate system is given by

$$M_{aw} = \begin{bmatrix} u_{xx} & u_{xy} & u_{xz} & 0 \\ u_{yx} & u_{yy} & u_{yz} & 0 \\ u_{zx} & u_{zy} & u_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ r_{0x} & r_{0y} & r_{0z} & 1 \end{bmatrix} \tag{6}$$

where the additional subscripts x, y, z indicate the appropriate components of the vectors in the world coordinate system. Each control point of patch (1) is transformed by M_{aw} as follows to give the patch geometry in the system of the algebraic surface

$$\mathbf{r}_a = \mathbf{r}_w\,[\,M_{aw}\,]^{-1} \tag{7}$$

where $\mathbf{r}_a$ and $\mathbf{r}_w$ are 1x4 row vectors of the control points in the local system of the algebraic surface and the world coordinate system, respectively.

4.2.2 Transformation 2

The original NURBS surface patch (after the above transformation) is now converted into the matrix, S, of its component rational polynomial patches by the use of a subdivision algorithm such as the Oslo algorithm outlined in Appendix VI

$$S = [\, S_{pq}(u,v) \,] \;\; where\; p = 0,\ldots m-k \; and\; q = 0,\ldots n-l \tag{8}$$

Each of the $S_{pq}(u,v)$ is a polynomial patch in the Bernstein-Bezier basis with the same degree in the u and v directions as the parent patch and provides the same geometrical B-spline surface in the appropriate span of u and v. Each rational Bezier patch can now be specified as

$$S_{pq}(u,v) = \sum_{i=0}^{k} \sum_{j=0}^{l} \mathbf{P}_{ij}^{pq}\, B_{i,k}(u)\, B_{j,l}(v) \tag{9}$$

where k, l are the degrees of the surface in the u and v directions, the (k+1)x(l+1) control points $\mathbf{P}_{ij}^{pq}$ given in the system of the algebraic surface and, for convenience, each surface patch can be considered to be parameterized in [0,1]x[0,1] without performing additional computation. Here $B_{i,k}(u)$ and $B_{j,l}(v)$ are Bernstein polynomials of degree k and l respectively.

Equation (9) representing the rational polynomial Bezier patch in the system of the algebraic surface and equation (5) representing the algebraic surface in its local system define the two intersecting surfaces in a simplified form that will provide an analytic representation of the intersection curve.

4.3 Representation of Intersection Curve

A point on the parametric patch $S_{pq}(u,v)$ in (9) is represented by a vector with homogeneous coordinates (x,y,z,w) which are bivariate polynomials in the variables u and v where u and v are in [0,1]

$$S_{pq}(u,v) = [\, x(u,v)\; y(u,v)\; z(u,v)\; w(u,v) \,] \tag{10}$$

A representation of the intersection of this rational polynomial parametric patch with the algebraic surface $H(x,y,z,w) = 0$ is obtained by substituting (10) in (5) to get an implicit polynomial equation

$$F_{pq}(u^m,v^n) = 0 \quad where \;\; u \in [0,1], v \in [0,1] \tag{11}$$

where the dependence u^m and v^n signifies that the maximum degree in u is m and in v is n and where the index p and q signifies that each polynomial subpatch of the original B-spline patch gives a separate equation representing the intersection in that region. This implicit equation describes, in general, an algebraic curve in the parametric space of the polynomial patch. Figure 4-1 shows the parametric domain of the B-spline patch with its component polynomial patches and the portion of intersection represented by (11). The restriction of u and v to be on the parametric space of the patch allows us to exploit the properties of the representation of this curve in the bivariate Bernstein basis as suggested in Patrikalakis [38], where such a method of representing a finite portion of a planar algebraic curve within a rectangular parallelogram is studied. The above implicit equation describes a planar algebraic curve of degree $m = kq$ and $n = lq$ in the two variables of interest where k and l are the degrees of the parametric patch in u and v and q is the degree of the algebraic surface. Such a curve is a <u>special</u> case of the general planar algebraic curve of degree m+n because no terms of the form u^{m+n} or v^{m+n} appear in (11). Algebraic curves of even modest degree can be very complex as outlined in a large body of literature on such curves [49, 33, 21, 18]. Solution of the intersection problem stated above, therefore, reduces to the discovery and description of all features of a planar algebraic curve in a finite rectangular domain and its subsequent mapping in three-dimensional space.

Derivation of equation (11) can be, for example, accomplished by converting the rational polynomial patch from its Bernstein form into its monomial form to allow easy multiplication of a number of bivariate polynomials resulting from the expansion of powers of x, y, z and w in $H(x,y,z,w) = 0$. This process leads to equation (11) expressed in the monomial form

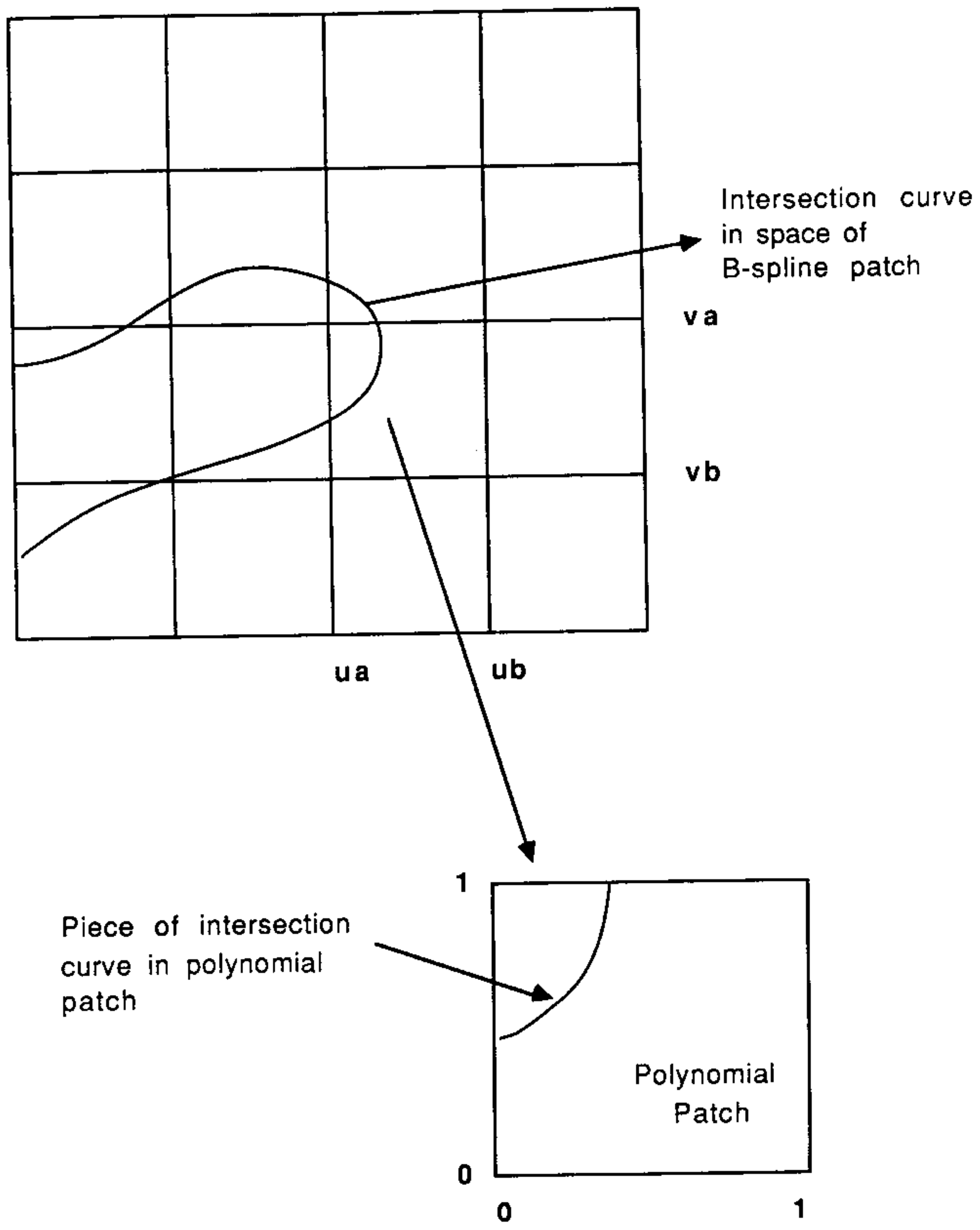
$$F(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} a_{ij}\, u^i v^j = 0 \quad \textit{where} \;\; u \in [0,1], v \in [0,1] \tag{12}$$

Here we have dropped the subscripts p,q and continue the derivation for one such typical component polynomial patch. In the case of the intersection problem at hand, where we are interested only in a finite portion of the curve within a rectangular parallelogram, it is expedient to transform the above monomial expansion into the Bernstein basis. The new coefficients w_{ij} are obtained by matrix multiplication

$$[\,W\,] = [\,B_m\,]\,[\,A\,]\,[\,B_n\,]^T \tag{13}$$

where $W = [\,w_{ij}\,]$ and $A = [\,a_{ij}\,]$ are (m+1)x(n+1) matrices and $[\,B_k\,]$ is a (k+1)x(k+1) transformation matrix defined in Appendix I. The inverses of the above matrices can be computed exactly as their elements are rational numbers. Superscript T denotes transpose.

Figure 4-1: Intersection curve of one polynomial patch



Algebraic curve (12) may now be expressed in the Bernstein basis as

$$F(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} B_{i,m}(u) B_{j,n}(v) = 0 \quad where\ u \in [0,1], v \in [0,1] \tag{14}$$

The numbers m, n and w_{ij} now define the geometry of the image of the intersection curve in the parametric u, v space of the patch (9). Curve (14), in general, has (m+1)(n+1)-1 degrees of freedom because one of the coefficients can be chosen arbitrarily without modifying the algebraic curve.

In the exceptional case, where $w_{ij} = 0$ for all i = 0,1...m, j = 0,1...n, (14) does not define a curve as the intersection of two surfaces. It, rather, points out that the rational polynomial patch coincides with a portion of the algebraic surface. When the coefficients of the algebraic surface are normalized as in (3), and floating point arithmetic is employed, the condition $| w_{ij} | \leq \varepsilon$, where ε is a small positive number, may be used to test for such an exceptional occurrence. As can be seen in Appendix II, the non-dimensional number $\varepsilon << 1$ is related to a distance tolerance, δ, for coincidence of the two surfaces, by the approximate equation $\varepsilon = \delta \, | \nabla H |$, where ∇ denotes gradient and ∇H is evaluated at a point S on the parametric surface assumed to be close to the algebraic surface.

In the sequel and for all other cases, it is convenient to assume that the coefficients w_{ij} are normalized using the maximum $|w_{ij}| \neq 0$, so that $|w_{ij}| \leq 1$ for all i and j. The above normalized w_{ij} will be used as the representation of the intersection curve for all further discussion. Using the above procedure, the Bernstein form of the algebraic curve may be coded symbolically and incorporated in an intersection program using the code writing capabilities of symbolic manipulation systems [36]. In general

$$w_{ij} = f_{ij} (\mathbf{P}^{pq}_{\mu\nu}, C_{def}) \tag{15}$$

where the $\mathbf{P}^{pq}_{\mu\nu}$ are the control points of surface (9) and C_{def} are the coefficients of the algebraic surface (5). The functions in (15) may be worked out for all pairs of required surfaces ie. for varying degree rational Bezier and algebraic surfaces, and directly incorporated into general intersection programs. Availability of explicit expressions for the coefficients w_{ij} (essentially summations of products) can be used to advantage, particularly in higher order cases, to get better numerical accuracy in the computation of w_{ij} in floating point arithmetic by employing sophisticated methods to sum series [29, 14]. The capabilities of symbolic manipulation systems can also be employed to factor terms and identify subexpressions in order to enhance efficiency [25].

4.4 Geometric Interpretation of Algebraic Curves in a Rectangular Parallelogram

It is now convenient to visualize (14) as the intersection of the explicit surface w = F(u,v) with the control plane w = 0. The above explicit surface, F(u,v), can now be recast in the following equivalent parametric tensor product Bezier patch form

$$\mathbf{T}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} \mathbf{T}_{ij}\, B_{i,m}(u)\, B_{j,n}(v) \quad where$$

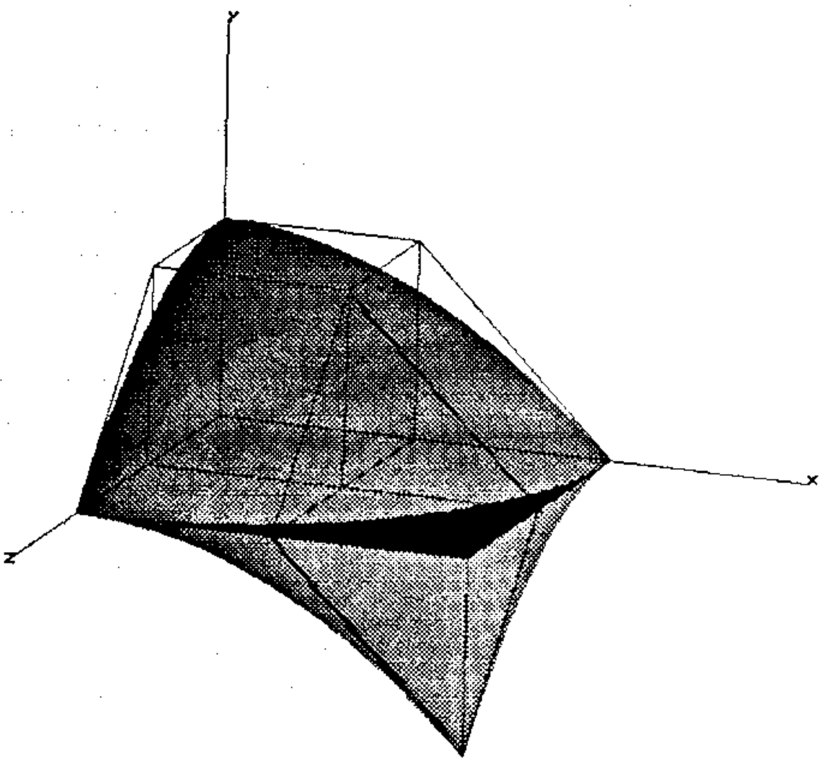
$$\mathbf{T} = [\, u\; v\; w\,]$$

$$\mathbf{T}_{ij} = [u_i' \;\; v_j' \;\; w_{ij}]$$

$$u_i' = \frac{i}{m} \; and \; v_j' = \frac{j}{n} \quad i = 0,1,...m \; and\, j = 0,1,...n \tag{16}$$

The coefficients w_{ij} introduced in (14) are the w coordinates of the control polyhedron vertices of the parametric surface defined by the above equation while the u and v coordinates of control polyhedron vertices are uniformly spaced in the range of the parametric variables. The above reformulation has been used by Geisow [21] in a study of surface interrogations. A similar representation method for algebraic curve segments within triangles has been studied by Sederberg [40]. Patrikalakis [38] uses the formulation (16) as a means to manipulate algebraic curves in a geometrically intuitive manner. Since the control polyhedron, $\mathbf{T}_{ij}$, provides an approximation to the geometry of the surface (16), we can get an approximation of the intersection curve (14) by intersecting the control polyhedron with the plane w = 0. The control polyhedron can be made piecewise planar by triangulating the rectangular grid of polyhedron vertices (faceting). Each of the triangles may then be intersected with the plane w = 0 to get a series of straight line segments which together form a piecewise linear approximation of the curve. The accuracy of this approximate intersection curve depends on the accuracy to which the control polyhedron approximates the surface. As is well known, the control polyhedron becomes an increasingly better approximation of the surface by the use of subdivision algorithms [13, 35]. A procedure of adding degrees of freedom to the polyhedron by surface subdivision at positions

Figure 4-2: Algebraic curve viewed as surface-plane intersection



where the approximation to the intersection curve is not very good, will allow the method to iterate until a certain level of accuracy of the intersection curve is reached. Unfortunately, under finite refinement of the polyhedron, the correct topological connectivity of the algebraic curve (14) will not in general be obtained. In particular, the singularities of the curve will not be traced correctly [18]. In addition, under finite refinement of the polyhedron, small internal spurious loops may also be present in the approximation of the algebraic curve using the above faceting method [38].

5. COMPUTATION OF SIGNIFICANT POINTS

5.1 Introduction

For finite refinement of the polyhedron, the approximate algebraic curve generated using the faceting method outlined in Section 4.4 does not, in general, exhibit the topological connectivity of the actual algebraic curve. In particular, portions of the curve near singularities cannot, in general, be approximated by plane-plane intersections and spurious small loops may also be introduced (see Figure 5-1). These problems were also identified in Section 3 where the motivation for the present method was presented.

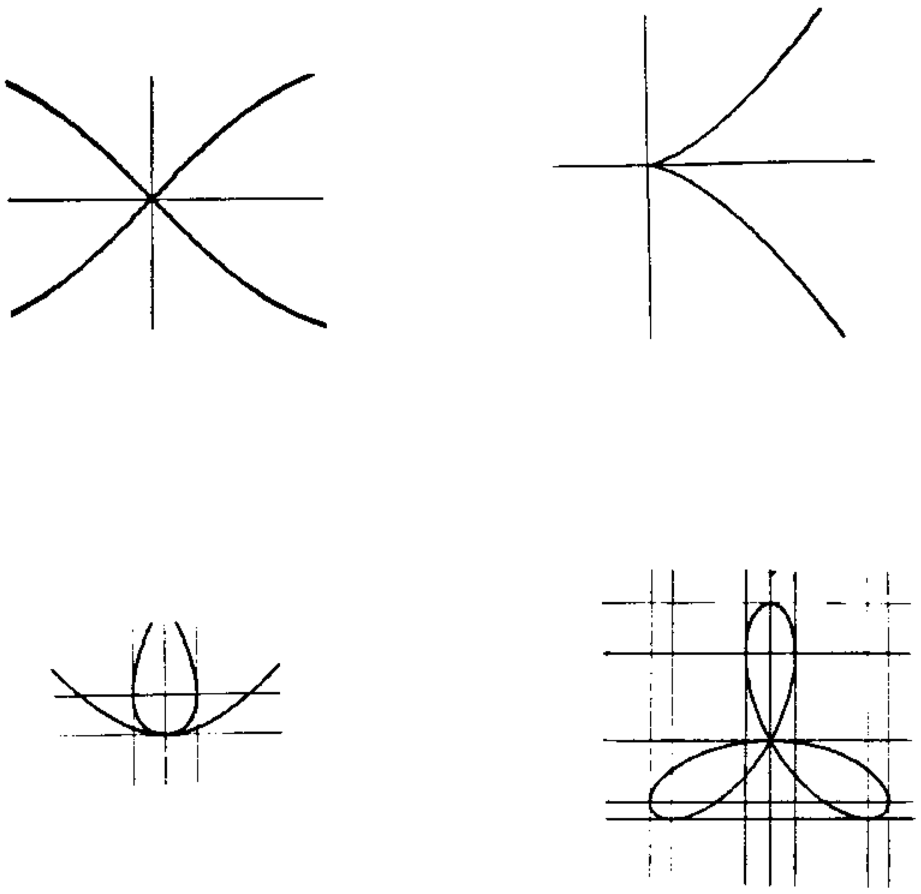
This suggests partition (splitting) of the control surface at turning and singular points so that the resulting pieces of the control surface represent **monotonic** curve segments that have no singularity or actual internal loops within their respective subdomains. This also implies that all curve complexity resides at corners of sub-patches. Once partition (splitting) of the surface has been carried out at all such significant points, the procedure of intersecting each facet of the polyhedron can be carried out and the resulting straight line segments can be connected together to reflect the true topological connectivity of the curve as described in Section 6 of this report. This reformulation is an important addition to existing subdivision and faceting algorithms and provides subdivision-based intersection solutions with the potential for extracting the true topology of the curve.

Singular points of an algebraic curve are defined by vanishing partial parametric derivatives ie. by the following three simultaneous equations

$$F(u,v) = F_u(u,v) = F_v(u,v) = 0 \tag{17}$$

An extensive discussion of singular points may be found in [49, 33, 18, 27, 7]. A summary of relevant material is given in Appendix III. Singular points are classified according to the number of partial derivatives beyond the first vanishing at such points. Points at which equation (17) is valid and at least one second order partial parametric derivative is non-zero are called double points. Points at which all second order partial derivatives vanish and at least one third order partial derivative is non-zero are called triple points (see Figure 5-1). In general, points on an algebraic curve at which all partial derivatives up to order m-1 vanish and at least one partial derivative of order m is non-zero are called singular points of multiplicity m.

Figure 5-1: Singularities in algebraic curves



Turning points are points on an algebraic curve at which the normal vector to the curve is parallel to the parametric u and v axes. For non-singular points, the unit normal vector is uniquely defined by $\mathbf{n} = \nabla F / |\nabla F|$ and, therefore, turning points on an algebraic curve may be defined by the following two sets of simultaneous equations

$$F(u,v) = 0 \tag{18}$$

$$F_u(u,v) = 0 \tag{19}$$

and

$$F(u,v) = 0 \tag{20}$$

$$F_v(u,v) = 0 \tag{21}$$

Equations (18) and (19) define the so-called v-turning points and equations (20) and (21) the u-turning points. Obviously, according to equation (17), points satisfying the requirements for the u <u>and</u> v turning points simultaneously, are not turning but singular points. Therefore, identification of all solutions of (18),(19) and (20),(21) also provides all singular points.

When the curve F(u,v) = 0 can be written as

$$F(u,v) = \prod_{i=1}^{M} F_i^{e_i}(u,v) = 0 \tag{22}$$

the curve is made up of component curves $F_i(u,v)$ with multiplicity e_i [49]. When $e_i \geq 2$ **all** points of the curve F(u,v) = 0 satisfy equations (17) to (21) and a modification of the method presented in this report is required. When $e_i = 1$ and when F_i does not represent a parametric line, our present method remains valid. When a parametric line is a component of the intersection curve, it should be factored out from the representation used to find turning and singular points because such lines form an infinity of turning points. Reliable methods to perform such factorisation are under investigation. In the present method, the border points caused by the parametric line will insure that the parametric line is included in the solution.

Beyond turning and singular points it is also convenient to identify **border** points a priori in order to allow reliable connection of the intersection curve with other parts of the curve outside the domain of interest. Border points are points of the curve at which at least one of the parametric variables take values equal to the borders of the rectangular parametric domain.

Border, turning and singular points are collectively called the **significant** points of the curve. In this work we are interested in the a priori computation of all significant points on or within a rectangular parallelogram.

5.2 Border Point Computation

The border points are located by solving a univariate polynomial in u for the v = 0 and v = 1 borders and a univariate polynomial in v for the u = 0 and u = 1 borders. Substituting these specific values for each border in the equation of the algebraic curve (14) and using the properties of the Bernstein basis we obtain

- v = 0 border

$$F(u,0) = \sum_{i=0}^{m} w_{i0} B_{i,m}(u) = 0 \tag{23}$$

- v = 1 border

$$F(u,1) = \sum_{i=0}^{m} w_{in} B_{i,m}(u) = 0 \tag{24}$$

- u = 0 border

$$F(0,v) = \sum_{j=0}^{n} w_{0j} B_{j,n}(v) = 0 \tag{25}$$

- u = 1 border

$$F(1,v) = \sum_{j=0}^{n} w_{mj} B_{j,n}(v) = 0 \tag{26}$$

The real roots of the above polynomials of degree m or n in the interval [0,1] can be found using a root-solving technique that directly exploits the properties of the Bernstein basis. Lane and Riesenfeld [32] outline such a technique for solving for the real roots of a polynomial in a given interval employing recursive binary subdivision and the variation diminishing property of this basis. For use with other standard polynomial root solvers available in commonly available mathematical libraries, [37, 36], the polynomial must be first converted into the monomial basis.

5.3 Turning and Singular Point Computation

5.3.1 Preliminary Remarks and Derivations

As stated earlier, the v-turning points satisfy (18) and (19) while the u-turning points satisfy (20) and (21). In matrix form, equations (18) and (19) become

$$[B_u^m]\,[W]\,[B_v^n]^T = 0 \tag{27}$$

$$[B_u^{m-1}]\,[W^u]\,[B_v^n]^T = 0 \tag{28}$$

where [W] is the matrix of w_{ij}'s of size (m+1)x(n+1) and [B_u^m] and [B_v^n] are the vectors of all Bernstein polynomials of degree m in u and degree n in v, respectively, and W^u is a matrix of size m x (n+1) with elements w_{ij}^u obtained from the Bernstein expansion of (19) and given by

$$w_{ij}^u = m\,(\,w_{i+1\,j} - w_{ij}\,) \qquad \textit{for } i = 0,1,...m{-}1 \textit{ and } j = 0,1,...n \tag{29}$$

All real solutions of the above simultaneous equations within the window of interest [0,1]x[0,1] are required for the determination of v-turning points. The conversion of (20) and (21) to matrix form and their solution follows a procedure similar to the derivation and solution of (27) and (28).

As stated earlier, singular points are located by solution of simultaneous equations (17), and hence, can be obtained as the common solutions of (18),(19) and (20),(21). Hence the problem reduces to the separate solution of the equations (18),(19) and (20),(21). Note that singularities of all orders satisfy the above equations and their multiplicities could be identified by additional computation of higher order derivatives (see Appendix III). A feature of the method developed in this report is that explicit a priori knowledge of the multiplicity of the point, based on higher order derivative evaluations, is not employed.

Equations (27) and (28) are two simultaneous bivariate polynomial equations, the real solutions of which in the region [0,1]x[0,1] are of interest. In this report, we exploit algebraic geometry techniques for reducing this problem to the solution of a univariate polynomial equation. The process involves elimination of one variable from a pair of bivariate polynomial equations [18]. This gives us a univariate polynomial equation which is, in general, of higher degree in the other variable than the degrees in u and v in the original equations. Such univariate polynomials can, subsequently, be solved without the help of initial estimates, an important advantage with respect to other iterative numerical techniques for the solution of systems of non-linear equations with an unknown number of roots within a given domain. All real solutions of this univariate equation give the set of values of one of the variables which possibly form one element of solution pairs of the simultaneous equations. These solutions are then back-substituted in the original equations such as (27) or (28) and further univariate polynomials (of generally much lesser degree) are solved to get values of the variable originally eliminated. From the solutions of the last univariate polynomial equation (ie. (27) or (28)), only the real solutions in [0,1] also satisfying **both** (27) and (28) are the turning point solutions.

The general theory of elimination deals with finding conditions which the coefficients of two univariate polynomial equations must satisfy in order to have common roots. A good discussion of elimination techniques is given in [49] and their use in computational geometry is discussed in [41, 18]. An extension of this technique can be used to eliminate one of the variables from two bivariate polynomial equations. Elimination theory with the polynomials expressed in the monomial basis is well developed. In what follows, we extend these techniques to the Bernstein basis. Working in the Bernstein basis is advantageous in geometric modeling applications, not only because of their geometric properties but also because of their computational properties. It has been observed, for example, that operations such as root computations on polynomials expressed in the Bernstein basis are numerically better conditioned as compared to similar computations in the monomial basis [38, 43]. More recently, it has been shown theoretically that the Bernstein basis is superior to the monomial basis for some common operations such as evaluation and root-finding [20]. Hence, we prefer to carry out this elimination step in the Bernstein basis directly (see Appendix IV for a more detailed derivation). Expressing (27) and (28) as polynomials in v we have

$$[A'][B_v^n]^T = 0 \quad where \quad [A'] = [B_u^m][W] \tag{30}$$

$$[B'][B_v^n]^T = 0 \quad where \quad [B'] = [B_u^{m-1}][W^u] \tag{31}$$

[A] and [B] are row vectors of size 1x(n+1) with elements defined by

$$a_j'(u) = \sum_{i=0}^{m} w_{ij} B_{i,m}(u) \qquad where\, j = 0,1, \ldots n \tag{32}$$

$$b_j'(u) = \sum_{i=0}^{m-1} w_{ij}^u B_{i,m-1}(u) \quad where\ j = 0,1,\ldots n \tag{33}$$

As pointed out in Appendix IV, we rewrite equations (30) and (31) so that the vectors $[B_v^n]$ contain only the polynomial part of the Bernstein coefficients without the leading combinatorial coefficients. In this manner, equations (30) and (31) become

$$\begin{aligned} &[A][L_v^n]^T = 0 \\ &[B][L_v^n]^T = 0 \end{aligned} \tag{34}$$

where the elements of row 1x(n+1) vectors [A] and [B] are defined by

$$\begin{aligned} a_j(u) &= \binom{n}{j} \sum_{i=0}^{m} w_{ij} B_{i,m}(u) \quad where\ j = 0,1,\ldots n \\ b_j(u) &= \binom{n}{j} \sum_{i=0}^{m-1} w_{ij}^u B_{i,m-1}(u) \quad where\ j = 0,1,\ldots n \end{aligned} \tag{35}$$

where

$$\binom{n}{j} = \frac{n!}{j!(n-j)!} \tag{36}$$

and the elements of the row 1x(n+1) vector $[L_v^n]$ by

$$l_{j,n}(v) = (1-v)^{n-j}\, v^j \quad where\ j = 0,1,\ldots n \tag{37}$$

We can now proceed to eliminate v from (34) using the technique discussed in Appendix IV leading to the following system of 2n homogeneous equations

$$\begin{aligned} &[\,D(u)\,].[L_v^{2n-1}]^T = [\,0\,] \ where \\ &d_{ij} = a_{j-i}(u) \quad for\ 0 \le i \le n-1\ and\ j \ge i \\ &\quad = b_{j-i+n}(u)\ for\ n \le i \le 2n-1\ \ and\ j \ge i-n \\ &\quad = 0 \qquad otherwise \\ &l_{j,2n-1}(v) = (1-v)^{2n-1-j}\, v^j \quad j = 0,1,2,\ldots,2n-1 \end{aligned} \tag{38}$$

We seek the set of values (u_i,v_i) such that the above system of equations is satisfied. A necessary and sufficient condition that the above system has non-trivial solutions is that $|D| = 0$, where $|\ |$

denotes determinant. Since all elements of the matrix D are polynomials in u, the determinant in this case is a polynomial in u of degree 2mn-n [18], called characteristic polynomial, which may be expressed in the monomial basis as follows

$$\sum_{i=0}^{2mn-n} c_i u^i = 0 \qquad (39)$$

This polynomial can then be solved to give the solution set u_i. For each of the u_i we can find the corresponding v_i coordinate by substituting in one of the equations (27) or (28) and solving a univariate polynomial equation in v of degree n. From these solutions, those which satisfy the other equation (from equations (27) and (28)) lead to the final solution set (u_i,v_i). Since the only solutions required are within the rectangle $u \in [0,1]$ and $v \in [0,1]$, this restriction can be used to expedite the computation at each stage by reducing the number of univariate polynomial equations to be solved for the other coordinate and the evaluations to find the points at which the other equation is also satisfied. Also note that, although the above elimination was carried out in the Bernstein basis, the resulting characteristic polynomial in u is expressed in the monomial basis since polynomial multiplication needed in direct determinant expansion is easier in this basis. This is a matter of concern, as we will see in the next section, where we discuss the conditioning of polynomial bases.

Equations (27) and (28) were solved by eliminating the variable v. We could instead have solved the equations by eliminating u. The degrees of the resulting polynomials in u or v are approximately of the same order and are equal when m = n. Eliminating v involves the expansion of a determinant of size 2n while eliminating u involves a size of 2m-1. For the case m = n, this involves the expansion of a slightly smaller determinant when we eliminate u. However elimination of u leads to an inherently ill-conditioned problem. This can be seen qualitatively from the fact that $dF = F_u du + F_v dv = 0$ along the curve and, hence,

$$\frac{du}{dv} = -\frac{F_v}{F_u} \qquad (40)$$

is very large in the neighborhood of a v-turning point so that a small change of v due to error will lead to a much larger change of u. This states that solutions for v-turning points should, preferably be performed by first eliminating v and solving for the u coordinate, unless an exact solution procedure is possible. If the other coordinate u is eliminated instead, then an error δv associated with the root-finding procedure for v will lead to large displacements δu of the corresponding u. Within floating point computation, the coefficients of the characteristic polynomial are perturbed quantities because of the process of expanding determinants of matrices whose coefficients are themselves polynomials. Accumulation of round-off error in the characteristic polynomial coefficients along with the inherent error accumulation characteristics of general root-solving techniques is bound to displace the values of v. For this reason, it is advantageous to eliminate v and solve in terms of u first. In the case of the u-turning point equation, we eliminate u and solve for v which, by analogous reasoning gives better conditioning in this case.

Expansion of determinants needed to obtain the coefficients of the characteristic polynomials is an inherently ill-conditioned problem [50]. It involves the multiplication of various elements of D, each of which is a polynomial in u. The multiplication of these polynomials involves summation of products (formed from coefficients of the polynomial elements of D) at each stage. This process has to be carried out a number of times before the final polynomial with its coefficients can be obtained. The errors at each stage caused by finite precision arithmetic and the degree reduction process will accumulate in each of the coefficients and may lead to substantial displacements of the roots of the characteristic polynomial. In this report, we investigate the use of symbolic computation as a means to reduce this kind of error for two reasons. First, since a number of coefficients of this matrix are equal to other coefficients, we expect a sizable amount of simplification to occur in the computation of these coefficients. If expansion is carried out symbolically, such simplifications can be easily performed leading to numerical error reduction compared to a direct numerical expansion. Second, the symbolic derivation of the characteristic polynomial in a symbolic system like MACSYMA [36] allows complete control over the various terms and symbols that combine to form the polynomial coefficients. That is, it allows each coefficient in the final polynomial (39) to be obtained as one continuous sum of products of the original Bernstein coefficients w_{ij} and w^{u}_{ij}. This control over the terms of the sum allows us to adopt sophisticated methods of summing a series of values which give a much smaller relative error on the sum than direct summation methods. Most importantly, following Kahan's method [29, 14], the relative error on the sum is independent of the number of terms N of the sum and linear in the machine precision. The above method offers a remarkable improvement over simple summation that is normally employed. See Appendix V for a statement of Kahan's algorithm and the error bounds for such a sum.

5.3.2 Determination of the Characteristic Polynomial

In this section, we describe two procedures for determining the polynomial coefficients c_i of (39) from the elements of [W] and [W^u] in (27) and (28) using the basic procedures outlined above.

5.3.2.1 Notation

The following discussion may be expressed concisely by defining the following arrays. The expansion of the determinant |D| in terms of its elements $a_i(u)$ and $b_j(u)$ can be written as

$$|D| = \sum_{k=1}^{kmax} \left(\prod_i a_i(u) \right) \left(\prod_j b_j(u) \right) \tag{41}$$

where *kmax* be the number of non-zero terms in the expansion of |D| in terms of its elements. The set i over which the first product is taken and the set j for the second product is a function of the particular term ie. of the index k. Let $e_i(u)$ be the combined set of polynomials $a_i(u)$ and $b_j(u)$ such that

$$e_i(u) = \begin{cases} a_i(u) & for\ i = 0,...n \\ b_{i-(n+1)} & for\ i = n+1,...2n+2 \end{cases} \tag{42}$$

By expanding the |D| symbolically we can find an unordered 2n tuple E_k for the k-th term in (41) which contains the indices of polynomials e_i involved in that term. There is one such tuple for each of the *kmax* terms and hence these *kmax* tuples of 2n numbers each may be combined into a matrix E of size kmax x 2n ie. each row contains the 2n indices for one term. E_{ik} is an element of the matrix that contains the ith member of the kth 2n tuple and product (41) becomes

$$\sum_{k=1}^{kmax} \prod_{i=1}^{2n} e_{E_{ik}}(u) \tag{43}$$

Let c_{ij} be the jth monomial coefficient of the ith polynomial $e_i(u)$ in (42) ie.

$$\begin{aligned} e_i(u) &= \sum_{j=0}^{m} c_{ij}\, u^j \quad for\ i = 0,...n \\ &= \sum_{j=0}^{m-1} c_{ij}\, u^j \quad for\ i = n+1,...2n+2 \end{aligned} \tag{44}$$

The coefficients c_{ij} are either the w_{ij} or w_{ij}^u or sums of those since here we define them as monomial coefficients of $e_i(u)$ while the original polynomials $a_i(u)$ and $b_j(u)$ were given in terms of the Bernstein basis.

The expansion of the determinant |D| in (41) leads to the characteristic polynomial (39). The products in (41) involves the product of polynomials of degree m or m-1 in u and which are combined in such a way as to give at most polynomials of degree 2mn-n in u [18]. On expansion this product of polynomials will contain terms each of which has a unique degree of "u" associated with it and is the product of 2n numbers each of which is a coefficient of one of the $e_i(u)$ ie. c_{ij}. Let us denote such a term by

$$T_{ijk} = \prod_{\mu=1}^{2n} c_{p_\mu q_\mu} \quad where\ p_\mu \in E_k\ and \sum_\mu q_\mu = i \tag{45}$$

where p_μ is an integer belonging to the kth row E_k of matrix E and i is the index specifying the degree of u in this term in (39), k is the index that specifies the term in (41) from which this term was extracted and j is the running index within the k-th term of (41) this term belongs to. The indices j and k need not be separated but they serve to clarify the parts in the problem and also to simplify discussion. The coefficients of the characteristic polynomial may then be expressed directly as summations of these T_{ijk} as

$$c_i = \sum_{k=1}^{kmax} \sum_{j=1}^{jmax(i,k)} T_{ijk} \tag{46}$$

Summation (46) extends over jmax(i,k)*kmax for each monomial coefficient and is a maximum when summing for the monomial of degree (2mn-n)/2 when this is an integer and for the integer part N of (2mn-n)/2 and N+1 otherwise.

5.3.2.2 Off-Line Symbolic Method

The expansion of the determinant |D| may be done symbolically ie. with the symbols for w_{ij} and w_{ij}^u instead of their specific values. Using tools available on symbolic systems like MACSYMA [36] we can extract each term T_{ijk} as defined above in (45) and directly code it into a FORTRAN program. The coefficients $c_{p_\mu q_\mu}$ are obtained symbolically in terms of the patch control points $\mathbf{P}_{ij}$ and the scalar coefficients C_{ijk} describing the algebraic surface as stated in (5) and coded into the above program. This allows the term T_{ijk} coded into the programs to be computed at run time given the patch control points and the scalar coefficients of the algebraic surface. These primary data determine w_{ij} as in (15) and in turn T_{ijk}. The coefficients of the characteristic polynomial may then be obtained at run time using the summation (46) modified to employ the summation technique due to Kahan [29]. However, the program size grows rapidly with the degrees of the intersecting surfaces, also reflecting increased computation. The rapid growth in the order of the summation required indicates that the intersection problem, when solved entirely in the symbolic manner just described, leads to a very large program generated by the symbolic manipulation system. This growth of program size points out the need to execute this process of having control over the summation in a semi-numerical, semi-symbolic way.

5.3.2.3 Semi-Symbolic/Semi-Numerical Method of Direct Summation

The symbolic method of determining the coefficients can be carried out in essentially the same way but without the growth of program size. It requires the computation of values for E_{ij} using the symbolic method but without the need to expand |D| and extract the symbols for each of the T_{ijk}. This process can now be done at run time. The important saving here is that the growth of program size is reduced to nearly constant with degree. This process of finding the coefficients of the characteristic polynomial and in determining the various T_{ijk} directly is best illustrated with pseudo-code

```
for k=1 to kmax
  {
  jmax(i,k) = 0   for all i = 0 to 2mn-n

  for j(1) = 0 to m
   for j(2) = 0 to m
    for j(3) = 0 to m
     ................

      for j(2n) = 0 to m
       {
       i = $\sum_{l=1}^{2n} j(l)$

       jmax(i,k) = jmax(i,k) + 1

       j = jmax(i,k)

       term(i,j,k) = $\prod_{l=1}^{2n} c_{E_{lk} j(l)}$
       }
  }
```

Here m is the degree in u and kmax is the total number of terms in the expansion of the determinant |D|. After obtaining the various T_{ijk} using the above scheme the coefficients of (39) are computed using Kahan's sum using (46).

5.3.3 Determination of Turning and Singular Points

The solution of equation (39) in u will lead to 2mn - n solutions $\{u_s\}$. Of these only the real solutions in the range [0,1] are of interest

$$\{u_s\} \quad s = 1,2,\ldots\mu \quad where\ \mu \le 2mn-n \tag{47}$$

Substitute each of the above $\{u_s\}$ in (27) to get

$$F(u',v) = \sum_{j=0}^{n} \left(\sum_{i=0}^{m} w_{ij} B_{i,m}(u') \right) B_{j,n}(v) = \sum_{j=0}^{n} w_j^u B_{j,n}(v) = 0 \quad where\ \ u' \in \{u_s\} \tag{48}$$

which is a univariate polynomial in v of degree n. This can be directly solved in the Bernstein basis (see next section) to get the set of solutions of corresponding v's $\{v_{st} : t = 1\ldots n\}$. Of these, only the real solutions in the range [0,1] are of interest, ie. $\{v_{st} : t = 1\ldots \nu(u_s)\}$, where $\nu(u_i) \le n$ is the number of real solutions of v for every solution u_s. These solution pairs are then substituted in (28) to remove those solutions that do not satisfy the second equation

$$F_u(u_s,v_t) = \sum_{i=0}^{m-1} \sum_{j=0}^{n} w_{ij}^u B_{i,m-1}(u_s) B_{j,n}(v_t) = 0 \tag{49}$$

This process is repeated for all solutions (47). Finally we get the entire solution set which satisfies both of the equations (27) and (28)

$$\{u_i,v_i\} \quad i = 1,2,\ldots p \tag{50}$$

where p is the number of v-turning points of the curve F(u,v) in [0,1]x[0,1].

The two bivariate polynomials F(u,v) = 0 and $F_v(u,v) = 0$ may be similarly solved to get the list of u-turning points. As discussed earlier, it is advantageous to eliminate u in this case to give

$$\{u_i,v_i\} \quad i = 1,2,\ldots q \tag{51}$$

where q is the number of u-turning points in this case.

The set of common solutions between the u and v turning points is then the set of singular points.

$$\{u_i,v_i\}, \quad i=1,2,\ldots r \quad where\ r \le \min(p,q) \tag{52}$$

Separate identification of singular points is not required in our method and the solution for the two types of turning points from (50) and (51) and the border points from (23) through (26) are merged and any duplicates removed. Such duplicates will occur at singular points where two points appear in the merged list once as a u-turning point and then as a v-turning point and when a point is both border, turning and/or singular at the same time. Two u or v values are considered the same if they are the same to the accuracy with which the roots of the characteristic polynomial are computed. The final list of significant points is denoted by

$$\{su_i, sv_i\} \quad i = 0,1,\ldots s \qquad where\ s \le 4mn + m + n - 1 \tag{53}$$

each element of which is at least one of the border, turning or singular points.

Before proceeding to use these significant points to split the surface into subpatches and trace the curve, we discuss the issues related to solving univariate polynomial equations in the next section.

5.4 Univariate Polynomial Root Computation

5.4.1 Introductory Remarks

This section describes the part of the intersection problem solution involving univariate polynomial root computation and discusses the issues involved in such numerical solutions in the monomial and Bernstein basis. In particular, issues of numerical stability and accuracy of computation are discussed. Computation of all real roots within a finite interval of a univariate polynomial is needed to determine the significant points of a planar algebraic curve. Specifically,

1. Border points require the solution of polynomials of degree m in u and degree n in v for each of the polynomial patches (9) as specified by (23) through (26).
2. To determine one coordinate of possible turning points requires the solution of a polynomial of degree 2mn-n in u (as in (39)) and of another polynomial of degree 2mn-m in v.
3. The solution for the other coordinate of a possible turning point requires univariate polynomial solutions of degree m in u and n in v.

The derivation of these univariate polynomial equations was explained earlier. While the solution for the border points and the back-substitution for the second coordinate of possible turning points involve solution of similar degree polynomials, the coefficients of the polynomials in the two cases may differ in accuracy. In the border point case, the univariate polynomial coefficients are the w_{ij} themselves and have incurred only a relatively small error in their computation. In the back-substitution case (48), the coefficients are obtained by evaluation of

Bernstein polynomials using the w_{ij} but at values of u or v that may be perturbed from its exact value. This perturbation is the error that may be incurred in the computation of one coordinate of possible turning points by evaluation and solution as in (39).

The quantities 2mn - n and 2mn - m, representing the degree of the turning point characteristic equations, are greater than or equal to m or n and, hence, the solution of the univariate polynomial for turning points forms the critical part of the determination of significant points. The degree of equations to be solved grows rapidly with the degree of the intersecting surfaces ie. k, l and q from which m and n are computed ($m = kq$ and $n = lq$). As the degree of the equation grows, the error involved in each evaluation of the polynomial using floating point arithmetic during the determination of roots grows linearly with the degree [20]. This implies that, when the coefficients are represented as accurately as the precision of the computer will allow, the accuracy in the evaluation and, therefore, solution of these equations will deteriorate with degree. A second, and possibly more serious, problem is significant root perturbation due to small perturbation of the coefficients. This may occur in the solution of any polynomial whose coefficients are not **primary data** but are the result of some intermediate computation [50, 14].

In order to appreciate the complexity of various intersection problems encountered in geometric modeling, we have compiled Table 5-1 illustrating the degrees of polynomials, the solution of which is *in principle*, needed to allow computation of significant points [12]. The degrees involved in the determination of turning points for the plane-biquadratic, plane-bicubic and quadric-torus (the torus being represented as a biquadratic) are of relatively low degree. The degree of the characteristic polynomial for some of the other cases like torus-torus(biquadratic) or torus-bicubic is relatively high. The high degree of the characteristic equation is, in part, caused by the high degree of the implicit representation of some of the commonly used rational polynomial parametric surfaces and the nature of the intersection problem itself.

Let us investigate qualitatively the inaccuracy involved in the computation of the coefficients of a characteristic polynomial such as (39). The inaccuracy may be conceptually separated into two parts. First, the inaccuracy involved in the computation of the w_{ij}, i.e. in the representation of intersection curve itself by a single algebraic equation. Second, the inaccuracy involved in the computation of the various terms T_{ijk} in (45) and in large scale series summations such as in (46). It is for this reason that we have investigated the use of Kahan's summation scheme [29, 14, 30] as a means to reduce the error incurred on each of the coefficients.

Surf A	Surf B	Curve Degree in Cartesian coordinates	Minimum curve exponents in parametric form $F(u^m, v^n) = 0$	Polynomial Degree	
				Border	Turning/ Singular
Plane	Torus	4	2,2	2	6
Quadric	Torus	8	4,4	4	28
Torus	Torus	16	8,8	8	120
Plane	E-cubic	3	3,1	3	3
Quadric	E-cubic	6	6,2	6	22
Torus	E-cubic	12	6,6	6	66
Plane	R-cubic	6	3,2	3	10
Quadric	R-cubic	12	6,4	6	44
Torus	R-cubic	24	12,8	12	184
Plane	P-cubic	18	3,3	3	15
Quadric	P-cubic	36	6,6	6	66
Torus	P-cubic	72	12,12	12	276

Note:
E-cubic = Ruled Patch with Rational Cubic Profile
R-cubic = Surface of Revolution with Rational Cubic Profile
P-cubic = Rational Bicubic

Table 5-1: Intersection Problem Complexity

The above discussion motivates a study of such polynomial equations with respect to two characteristics

1. The accuracy to which the roots can be computed if the polynomial coefficients are known to the full precision of the machine.
2. The stability of the roots relative to perturbation in the coefficients.

The first criterion has been studied extensively for the purposes of stopping root-finding algorithms [50]. However, no a priori guarantees of a specific accuracy on the roots may be obtained in general. Well coded root-finding techniques accept an approximation of a root as the best approximation, when an upper bound of the round-off error incurred in the evaluation of the polynomial is comparable in magnitude to the value of the polynomial [2]. In our work, we have evaluated some of the commonly available root-finding algorithms in connection with high degree polynomials to estimate the accuracy to which the roots may be computed.

The second criterion has been investigated in some recent work by Farouki and Rajan [20]. They have compared the relative performance of the two well known polynomial bases, the monomial (power) and Bernstein bases and shown theoretically that the Bernstein basis is better conditioned for the evaluation of polynomials and in the determination of roots. The above theoretical results corroborate observations made in the use of Bernstein polynomials for finding roots of univariate polynomials of high degree arising from intersections of planar algebraic and parametric curves [43, 38]. Let us consider a polynomial, P, like that of (39), of degree n expressed in the monomial basis and where we are interested in its behavior in a finite region say [0,1]

$$P(u) = \sum_{i=0}^{n} a_i u^i \quad u \in [0,1] \tag{54}$$

Let Q be the polynomial P expressed in the Bernstein basis

$$Q(u) = \sum_{i=0}^{n} b_i B_{i,n}(u) \quad u \in [0,1] \tag{55}$$

Evaluation of the conditioning of polynomials P and Q can be performed using condition numbers. The conditioning of a given operation may be defined as the sensitivity of the output values to some (random) infinitesimal perturbation of its input parameters. A high value of the condition number implies that the problem is ill-conditioned with respect to that operation, while a low value indicates that it relatively well-conditioned. We consider the condition numbers for the two most common operations encountered in connection with polynomials, i.e. evaluation and root location. Here, the a_i and b_i are the input parameters for two cases. For the case of evaluation, the result of Horner's algorithm to evaluate polynomial P at u and de Casteljau's algorithm to evaluate polynomial Q at u are the output parameters. In the case of the root location, the output parameters are the values of the roots themselves. The condition number for evaluation is given by

$$C(u_0) = \sum_{i=0}^{n} |c_i \phi_i(u_0)| \tag{56}$$

and for root-location by

$$C(u_r) = [\frac{m!}{|R^m(x_r)|}\sum_{i=0}^{n} | c_i\, \phi_i(u_r) |]^{1/m} \tag{57}$$

where c_i are the coefficients of some polynomial R(u) expressed in any basis $\phi_i(u)$ and R^m denotes m-th order derivative. Hence the above formulae apply for evaluations both in the monomial and Bernstein bases. Farouki and Rajan [20] show that these condition numbers are always larger in the monomial basis than in the Bernstein basis.

5.4.2 Illustrative Examples

We illustrate the above results by the use of three typical polynomial examples.

$$P1(u) = \prod_{k=1}^{20} (u - \frac{k}{20}) = \sum_{i=0}^{20} a_i u^i$$

$$P2(u) = \prod_{k=1}^{10} (u - \frac{1}{k})$$

$$P3(u) = [\prod_{k=1}^{8} (u - \frac{1}{k})](u - \frac{9}{10})^2 \tag{58}$$

Polynomial P1 is of degree 20 and has uniformly distributed roots in the range [0,1]. P2 is of degree 10 with simple roots that are clustered near zero. P3 is of degree 10 with one double root and a number of roots clustered near zero.

$$P4(u) = \prod_{k=1}^{20} (u - \frac{k}{20}) + \varepsilon a_{19} u^{19} \; and \; \varepsilon = \frac{2^{-23}}{210}$$

$$P5(u) = \sum_{k=1}^{20} c_k B_{k,20}(u)$$

$$P6(u) = \sum_{k=1}^{20} c_k B_{k,20}(u) + \varepsilon c_{19} B_{19,20}(u) \tag{59}$$

P4 is a minor perturbation of polynomial P1 with modified coefficient of the monomial of degree 19 [50, 20]. P5 is polynomial P1 expressed in the Bernstein basis. P6 is a minor perturbation of P5 in the coefficient of the nineteenth Bernstein polynomial. The coefficients of P5 are obtained to the same relative accuracy as the coefficients of P1 ie. the coefficients of P5 are not the result of a floating-point transformation from the monomial to the Bernstein basis but are obtained by converting the rational monomial coefficients into the Bernstein basis using rational arithmetic and expressing it in floating point arithmetic with one final division. This allows both the coefficients of P1 and P5 to be obtained to 16 decimal digit precision.

We have investigated two existing techniques for computing the roots of polynomials expressed in the monomial basis and one existing subdivision technique for polynomials expressed in the Bernstein basis. Specifically

1. Grant and Hitchins [23, 37]
2. The algorithm due to Jenkins [28, 36].
3. A subdivision algorithm based on the variation diminishing property of Bernstein basis due to Lane and Riesenfeld [32], see Appendix VII.

The above three techniques have been applied to the polynomials described above in double precision arithmetic (16 decimal digits) and the result of the solutions of these equations is given in Tables 5-2 through 5-13.

No:	Real	Imaginary
1	+1.00000102425157550e+00	0.00000000000000000e+00
2	+9.49987221458424820e-01	0.00000000000000000e+00
3	+9.00071295615115630e-01	0.00000000000000000e+00
4	+8.49759777000832200e-01	0.00000000000000000e+00
5	+8.00532848502709590e-01	0.00000000000000000e+00
6	+7.49140982568724650e-01	0.00000000000000000e+00
7	+7.00994924871775840e-01	0.00000000000000000e+00
8	+6.49148429906997930e-01	0.00000000000000000e+00
9	+6.00544614557426480e-01	0.00000000000000000e+00
10	+5.49757675291008320e-01	0.00000000000000000e+00
11	+5.00071323304214580e-01	0.00000000000000000e+00
12	+4.99999999999259090e-02	0.00000000000000000e+00
13	+4.49991128059475820e-01	0.00000000000000000e+00
14	+3.99997594172206260e-01	0.00000000000000000e+00
15	+3.50001449686100440e-01	0.00000000000000000e+00
16	+2.99996778492624l0e-01	0.00000000000000000e+00
17	+2.50000033693795430e-01	0.00000000000000000e+00
18	+1.49999999833262320e-01	0.00000000000000000e+00
19	+1.00000000009829800e-01	0.00000000000000000e+00
20	+1.99999999367336330e-01	0.00000000000000000e+00

Table 5-2: Roots of P1 using the Grant-Hitchins's method

It is important to note that in the following comparisons the relative error in the coefficients of polynomials P1 and P5 remains the same and is not the result of floating point conversion of coefficients from one basis to the other. Tables 5-2 to 5-4 show the results of using the first two methods on P1 and the third on P5. At worst, only *three* decimal digits of precision were

No:	Real
1	0.0500000002057265
2	0.0999999958524556
3	0.1500000409123849
4	0.1999997403294993
5	0.2500011751969473
6	0.2999960479663556
7	0.3500097453868570
8	0.3999847917920879
9	0.4500010172002592
10	0.5000796096278225
11	0.5497202627798882
12	0.6006023130416013
13	0.6491034551630523
14	0.7009991584238860
15	0.7491706328774203
16	0.8004962445992709
17	0.8497830857618643
18	0.9000629083128389
19	0.9499888814177023
20	1.0000008931520800

Table 5-3: Roots of P1 using the Jenkin's method

obtained in all three cases when the coefficients were all obtained correct to 16 decimal digits. Tables 5-5 and 5-6 shows the results of the first two methods on P4, namely the perturbation of P1 in the monomial basis. Table 5-7 shows the results of the third method on P6, the perturbation of P5. A more disturbing result is observed in the case of P4 in Tables 5-5 and 5-6 where roots greater than 0.4 are significantly displaced from the actual roots and this displacement occurs in both the real and imaginary axes. Thus a problem in which only real solutions are meaningful will lead us to ignore the complex solutions leading to a complete loss of a possible root. On the other hand, the perturbation of P5 ie. P6 does not loose any of the roots and the accuracy of the new roots is similar to that which was obtainable from the unperturbed equation. This

No:	Real
1	+5.000000000001391e-02
2	+9.999999999632369e-02
3	+1.500000002714119e-01
4	+1.999999932412781e-01
5	+2.500000641530011e-01
6	+3.000000121872668e-01
7	+3.4999943180416797e-01
8	+4.0000587653057960e-01
9	+4.4996686238887777e-01
10	+5.0012445838000304e-01
11	+5.4967148049980004e-01
12	+6.0065186513578715e-01
13	+6.4905327781006326e-01
14	+7.0105308666128412e-01
15	+7.4911674334829860e-01
16	+8.0053841943393196e-01
17	+8.4975894174666892e-01
18	+9.0007093875592782e-01
19	+9.4998776840550751e-01

Table 5-4: Roots of P5 using the subdivision method

essentially illustrates the observation based on the condition numbers that the Bernstein basis is inherently stable for the same relative accuracy on the coefficients. Tables 5-8 to 5-10 show the computed roots of polynomial P2 and Tables 5-11 to 5-13 the roots of polynomial P3 which has multiple roots with low order multiplicity. These examples again illustrate the relatively low accuracy obtainable in the roots of polynomials of degree even as low as ten. It is, however, worth noting the very small reduction in accuracy for the double root. Double roots are among the most common multiple root, since, as it turns out, double roots occur in the case of symmetric curves.

A conclusion which may be drawn from the above tables, also based on the theoretical results reported in [20], is that all computations are best consistently carried out in the Bernstein basis.

No:	Real	Imaginary
1	+9.751220637889678900e-01	-9.701647749973664800e-02
2	+9.751220637889678900e-01	9.701647749973664800e-02
3	+6.996202484183219200e-01	-1.259420414598087000e-01
4	+6.996202484183219200e-01	1.259420414598087000e-01
5	+1.042345417988604000e+00	0.000000000000000000e+00
6	+5.896872523576614500e-01	-8.262373895835258300e-02
7	+5.896872523576614500e-01	8.262373895835258300e-02
8	+8.365373959858483900e-01	1.406311495660580000e-01
9	+8.365373959858483900e-01	-1.406311495660580000e-01
10	+5.047585441046437600e-01	-3.219757086401624600e-02
11	+5.047585441046437600e-01	3.219757086401624600e-02
12	+4.999999999992590300e-02	0.000000000000000000e+00
13	+4.458561577215673800e-01	0.000000000000000000e+00
14	+4.003610666720891700e-01	0.000000000000000000e+00
15	+3.499863008179522400e-01	0.000000000000000000e+00
16	+3.000000239030295400e-01	0.000000000000000000e+00
17	+2.500000303401401300e-01	0.000000000000000000e+00
18	+1.499999998334959800e-01	0.000000000000000000e+00
19	+1.000000000098329400e-01	0.000000000000000000e+00
20	+1.999999993629357000e-01	0.000000000000000000e+00

Table 5-5: Roots of P4 using the Grant-Hitchins's method

Further, the degradation in the accuracy of the roots as the degree of the polynomial increases suggests the need for extended precision floating point arithmetic in the computation of the coefficients and the roots in order to obtain a relatively high accuracy in the root estimation.

In our current implementation, allowing computation of turning points for plane-biquadratic and plane-bicubic intersections we employ a method based on the monomial basis which makes use of the Grant-Hitchins algorithm [23, 37]. It is possible to convert the monomial coefficients to the Bernstein basis and proceed with the solution of the polynomial using a subdivision method. But the error incurred in the transformation carried out in floating point arithmetic could cause further displacement of the coefficients. Hence, it is preferable to obtain the

No:	Real	Imaginary
1	0.05000000020572652	
2	0.09999999585245565	
3	0.15000004091230140	
4	0.19999974034627820	
5	0.25000117145803900	
6	0.29999639665930630	
7	0.34999458466946010	
8	0.40034783136596030	
9	0.44586618154688920	
10	0.50475682891518600	+0.03220001973961333
11	0.50475682891518600	-0.03220001973961333
12	0.58968832685088290	+0.08262316446178804
13	0.58968832685088290	-0.08262316446178804
14	0.69962004060083890	+0.12594148323243860
15	0.69962004060083890	-0.12594148323243860
16	0.83653719431246330	+0.14063098832314130
17	0.83653719431246330	-0.14063098832314130
18	0.97512196235175660	+0.09701642568458171
19	0.97512196235175660	-0.09701642568458171
20	1.04234535688178800	

Table 5-6: Roots of P4 using the Jenkins's method

characteristic equation in the Bernstein basis directly in order to exploit the properties of the Bernstein basis fully.

No:	Real
1	+5.0000000000001787e-02
2	+9.9999999999620079e-02
3	+1.5000000001863876e-01
4	+1.9999999933904425e-01
5	+2.5000000605425942e-01
6	+3.0000000520937474e-01
7	+3.4999940539906325e-01
8	+4.0000598746128155e-01
9	+4.4996640782677112e-01
10	+5.0012553793065515e-01
11	+5.4966944353509732e-01
12	+6.0065119917313582e-01
13	+6.4906958163050656e-01
14	+7.0099883479704003e-01
15	+7.4920884571505467e-01
16	+8.0044243585810600e-01
17	+8.4982056452637660e-01
18	+9.0005082666199299e-01
19	+9.4998948992984507e-01

Table 5-7: Roots of P6 using the subdivision method

No:	Real	Imaginary
1	+1.0000000000000070e+00	0.0000000000000000e+00
2	+4.9999999999998640e-01	0.0000000000000000e+00
3	+3.3333333333326600e-01	0.0000000000000000e+00
4	+2.5000000000025720e-01	0.0000000000000000e+00
5	+1.9999999999980610e-01	0.0000000000000000e+00
6	+9.9999999999914390e-02	0.0000000000000000e+00
7	+1.6666666666627120e-01	0.0000000000000000e+00
8	+1.4285714285724047e-01	0.0000000000000000e+00
9	+1.2499999999908540e-01	0.0000000000000000e+00
10	+1.1111111111154180e-01	0.0000000000000000e+00

Table 5-8: Roots of P2 using the Grant-Hitchins's method

No:	Real
1	0.0999999999999846
2	0.1111111111111460
3	0.1249999999998841
4	0.1428571428574500
5	0.1666666666662729
6	0.2000000000002548
7	0.2499999999999152
8	0.3333333333333417
9	0.5000000000000068
10	0.9999999999999978

Table 5-9: Roots of P2 using the Jenkins's method

No:	Real
1	+1.0000000000003685e-01
2	+1.1111111111091033e-01
3	+1.2500000000046720e-01
4	+1.4285714285653840e-01
5	+1.6666666666713979e-01
6	+1.9999999999977171e-01
7	+2.5000000000006668e-01
8	+3.3333333333332192e-01
9	+5.0000000000000099e-01
10	+9.9999999999999999e-01

Table 5-10: Roots of P2 using the subdivision method

No:	Real	Imaginary
1	+9.00000000000487330e-01	-2.94109250799949350e-07
2	+9.00000000000487330e-01	2.94109250799949350e-07
3	+7.99999999999845860e-01	0.00000000000000000e+00
4	+6.99999999997237910e-01	0.00000000000000000e+00
5	+6.00000000003355310e-01	0.00000000000000000e+00
6	+4.99999999998116720e-01	0.00000000000000000e+00
7	+4.00000000000533260e-01	0.00000000000000000e+00
8	+2.99999999999935170e-01	0.00000000000000000e+00
9	+1.00000000000000160e-01	0.00000000000000000e+00
10	+2.00000000000000910e-01	0.00000000000000000e+00

Table 5-11: Roots of P3 using the Grant-Hitchins's method

No:	Real
1	0.0999999999999993
2	0.2000000000000084
3	0.2999999999985682
4	0.4000000000102053
5	0.4999999999680324
6	0.6000000000569322
7	0.6999999999361859
8	0.8000000000503784
9	0.8999990963351575
10	0.9000009036445324

Table 5-12: Roots of P3 using the Jenkins's method

No:	Real
1	+9.9999999999999889e-02
2	+2.0000000000000071e-01
3	+2.9999999999921295e-01
4	+4.0000000000003407e-01
5	+5.0000000000000000e-01
6	+5.0000000000000000e-01
7	+6.0000000000005498e-01
8	+6.9999999999997746e-01
9	+8.0000000000000455e-01
10	+9.0000095367431164e-01
11	+9.0000095367431164e-01

Table 5-13: Roots of P3 using the subdivision method

6. CURVE TRACING

6.1 Partition at Significant Points

This section describes a technique of tracing an algebraic curve given the Bernstein representation of the curve and the set of significant points of the curve as discussed earlier. This technique is directly relevant to the computation of the intersection of an algebraic and a rational polynomial parametric surface patch because such intersection is an algebraic curve in the parametric space of the patch. The successful tracing of such an algebraic curve, with correct connectivity between its various branches, can then be mapped via the parametric surface equation into the three-dimensional space of the problem to give the required intersection curve. The Bernstein representation of an algebraic curve allows the curve to be viewed as the intersection of parametric surface (16) with the plane w = 0 (see Section 4)

$$\mathbf{T}_{m,n}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{T}_{ij} B_{i,m}(u)\, B_{j,n}(v) = 0 \tag{60}$$

As described earlier in Section 5 we use the significant points of the algebraic curve to trace it in a piecewise manner. We first split (partition) the surface (60) in such a manner that the u and v coordinates of all significant points define the borders of these subpatches and the actual significant points lie only at corners of the subpatches. It is possible to split a patch into its subdomain patches along parametric lines using the Oslo subdivision algorithm for B-spline curves and surfaces (see Appendix VI). The Oslo algorithm allows non-uniform refinement of the polyhedron by adding new knots to the initial knot vector of a B-spline curve or surface. By adding knots of multiplicity m+1 and n+1 at specific u and v values it is possible to extract the subpatch corresponding a given parametric subdomain $[u_a, u_b]\mathrm{x}[v_c,v_d]$.

Let the list of significant points obtained in the earlier section be denoted by

$$\{su_i, sv_i\} \;\; i = 0,1,\ldots s \quad \textit{where } s \le 4mn + m + n - 1 \tag{61}$$

For the purposes of splitting the surface into sub-surfaces according to the criteria mentioned above, we construct separate lists of u and v values at which there is at least one significant point. The list given in (61) could in general have more than one occurrence of the same u or v values at different significant points. These u and v ordinates are ordered by increasing magnitude and then any duplicates are removed. Values of u and v are considered duplicate if they differ by a quantity the absolute value of which is less than ε_{sort}. This reduced set of significant ordinates are

$$\begin{aligned}&\{su_i\},\ i = 0,\dots p\\ &\{sv_j\},\ j = 0,\dots q \ \ \textit{where } p,q \le s\end{aligned} \tag{62}$$

This set of significant u and v ordinates can then be arranged into the following rectangular parametric domains at the borders of which the original control surface (60) must be split

$$[(su)_i,(su)_{i+1}]\ x\ [(sv)_j,(sv)_{j+1}] \quad \textit{for } i = 0,\dots p-1 \quad \textit{and } j = 0,\dots q-1 \tag{63}$$

Let us denote one such rectangular domain by $[u_a, u_b]x[v_c, v_d]$. In order to obtain the subpatch bounded by the above parameters, we insert m+1 interior knots at u_a and u_b and n+1 interior knots at v_c and v_d into the knot vector of the Bezier surface given by (60). The new knot vector is thus specified by

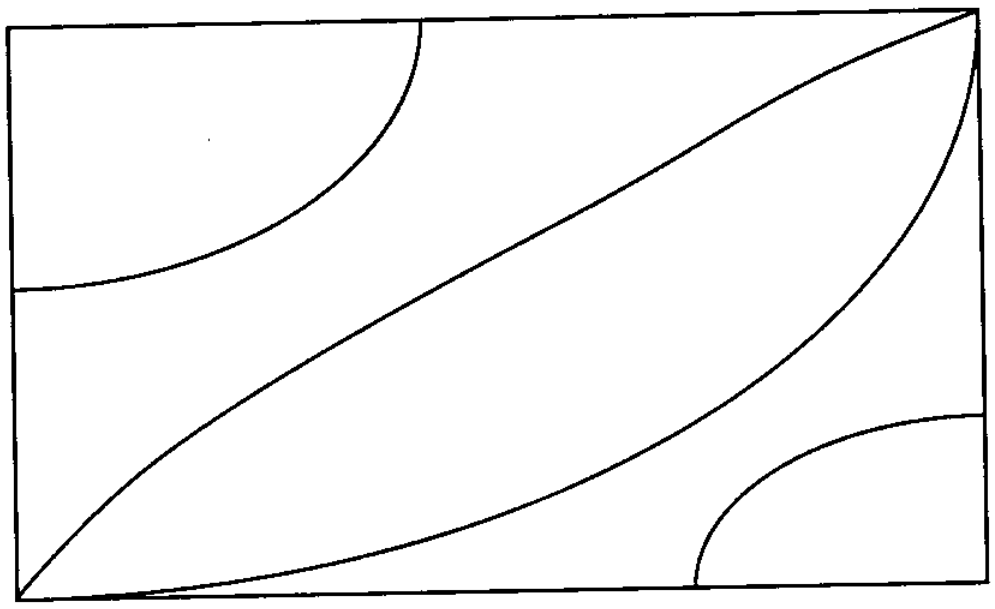
$$\begin{aligned}\{\,u_i\,\}' &= 0 && i = 0,\dots m\\ &= u_a && i = m+1,\dots 2m+2\\ &= u_b && i = 2m+3,\dots 3m+4\\ &= 1 && i = 3m+5,\dots 4m+6\\[1ex] \{\,v_i\,\}' &= 0 && i = 0,\dots n\\ &= v_c && i = n+1,\dots 2n+2\\ &= v_d && i = 2n+3,\dots 3n+4\\ &= 1 && i = 3n+5,\dots 4n+6\end{aligned} \tag{64}$$

If any of the values, u_a, u_b, v_c, v_d happen to be on one of the borders of the original Bezier patch, ie. have values 0 or 1, then such values need not be inserted into the knot vector of the original surface. The original surface may be split using the finer knot vector given by (64) and the Oslo algorithm to obtain a geometrically identical set of B-spline surfaces. The details of such a splitting procedure for rational B-spline surfaces are outlined in Appendix VI. Let us denote the subpatch corresponding to sub-domain $[u_a,u_b]x[v_c,v_d]$ by

$$\mathbf{P}_{m,n}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathbf{P}_{ij}\,B_{i,m}(u)\,B_{j,n}(v) \tag{65}$$

This subpatch now provides the representation of the intersection curve in the subdomain $[u_a,u_b]x[v_c,v_d]$. The curve does not contain singular, turning or border points in the interior of this subdomain. However, this subdomain may contain more than one segment of the algebraic curve (see Figure 6-1), each of which is **monotonic**.

Figure 6-1: Subpatches with their intersections



6.2 Tracing Curve Segments Represented by One Subpatch

The tracing of the curve segments represented by one subpatch is based on a faceted approximation of the subpatch and representation of the curve segments as intersections of this faceted approximation with the control plane as explained earlier. The particular faceted approximation chosen is based on the polyhedron formed by a triangulation of the control polyhedron net of the Bezier subpatch. Before we proceed with the specifics of triangulation and intersection of triangular facets with the control plane, we describe some preliminary computations which should be carried out ahead of time in order to efficiently eliminate non-intersecting subpatches.

6.2.1 Elimination of Empty Subpatches

It is possible (and, in fact, very frequent) that a given subpatch, produced by the splitting at significant points does not contain a portion of the algebraic curve. It is also possible that the control polyhedron intersects the control plane, but the underlying control surface itself does not intersect the plane. This occurs when the control surface is close to the plane. Since we have computed all turning points and split the surface at such points, we are guaranteed that any intersection of polyhedron occurring in its interior exclusively does not contribute to an intersection in our case. Hence we screen each subpatch for proper intersections and proceed with the triangulation and intersection with a plane only if the following conditions hold true

1. There is at least one sign change among the w_{ij}'s. A sign change is identified when w_{ij} exist in more than one of the following intervals of the real w axis $[-1, -\varepsilon)$, $[-\varepsilon,\varepsilon]$,$(\varepsilon,1]$, where ε is a small positive number, denoting the accuracy used to decide whether a w_{ij} coordinate may be considered to lie on the control plane. Henceforth, we refer to this ε as ε_{plane}.
2. There is at least one sign change along the edge (border) control points of the subpatch. This condition is required so as to prevent attempting to intersect subpatches whose control polyhedra at some finite level of subdivision still intersect the plane (usually in small loops) but the underlying surfaces themselves do not intersect. As stated earlier, we are guaranteed not to have small loops within our sub-domains. In fact, curve segments within our sub-domains are **monotonic** and start and end at **distinct** borders. This is the reason why we test edge control points for at least one sign change.
3. We also recognize cases where intersection segments are entirely on one or more of the edges of the subpatch.

A subpatch is first processed to verify the above conditions and the tracing procedure is attempted only when these conditions are true.

6.2.2 Preliminary Subdivision

A subpatch which has a proper intersection is, at first, a Bezier surface represented as a B-spline surface and, hence, contains no internal knots. The control polyhedron of such a surface has (m+1)x(n+1) vertices. If, at the corners of this surface, parametric partial derivatives vanish in either direction to an order less than the order of the surface in the respective directions, the control points will in turn lie on the plane. This will create degenerate triangle-plane intersections and thus cause sufficient loss of intersection information for the entire surface so that possibly no proper intersections are obtained when all facets are intersected with the plane. Such a condition is not a rare occurrence for the problem being solved since corners of the subpatches we are dealing with may contain singular points. Turning and singular points may lead to vanishing of parametric derivatives of various orders at the corners. As a way of increasing the intersection accuracy from the first step and isolating the effect of vanishing derivatives at different corners, we introduce m+2 and n+2 uniformly spaced internal knots in the u and v directions, respectively, and find the corresponding new set of control points of the new, geometrically identical, B-spline surface using the Oslo algorithm. In the u-direction, the original m+1 control points now become 2m+3 control points and in the v-direction n+1 become 2n+3, thus separating the effect of each corner from the effect of the other corners. The additional knots in the knot vector of the subpatch are

$$u_i = u_a + \frac{i}{m+3}(u_b - u_a) \quad i = 1,2,...m+2$$

$$v_i = v_a + \frac{j}{n+3}(v_b - v_a) \quad i = 1,2,...n+2 \tag{66}$$

This preliminary subdivision of the subpatches also brings the polyhedron closer to the surface which makes the approximation of the algebraic curve more accurate right at the start of the iteration.

6.2.3 Triangulation and Computation of Individual Triangle/Plane Intersections

The control polyhedron of the B-spline surface after the above preliminary subdivision is triangulated in one of many possible ways. The particular scheme used here is to subdivide each three dimensional quadrilateral formed by the control points into two triangles by joining control vertices $\mathbf{T}_{ij}$ and $\mathbf{T}_{i+1\ j+1}$. This induces a bias in the triangulation which is maintained consistently throughout the triangulation of all subpatches. The effect of such a bias in the reliability of our method should be investigated. An alternate method of triangulation of a parametric tensor product surface patch can be found in [46]. The above triangulation may be expressed by the following two sets of coordinate triplets over all the control points

$$\begin{aligned} T_{ij0} &= [\, \mathbf{T}_{ij},\ \mathbf{T}_{i+1j},\ \mathbf{T}_{i+1\,j+1} \,] \\ T_{ij1} &= [\, \mathbf{T}_{ij},\ \mathbf{T}_{ij+1},\ \mathbf{T}_{i+1\,j+1} \,] \end{aligned} \qquad (67)$$

Each quadrilateral consists of two triangles each denoted T_{ijk}, where k = 0 or 1 and is the index specifying one of the two triangles whose common edge extends from $\mathbf{T}_{ij}$ and $\mathbf{T}_{i+1\ j+1}$. Each of the triangles T_{ijk} is then intersected with the plane to get a piece of the piecewise linear approximation of the algebraic curve. The results of this intersection are retained to a high level of detail so that they may be used later to directly form a connected sequence of these intersection segments.

The procedure of intersecting the triangles with the plane is the most repeated computation and grows as O(pq) where p, q are the number of control point rows of the subpatch in the u and v directions. Since the polyhedron is iteratively refined in order to satisfy a certain level of accuracy, this step is also repeated in every iteration and forms the fundamental, computation intensive, intersection stage. Improvement in the efficiency of this stage will consequently drastically affect the performance of our algorithm. Each triangle-plane intersection for one iteration can be thought of as entirely independent of other triangle-plane intersections and could, in the future, be parallelized in such a way that each iteration would only take the time now taken to intersect one triangle.

The intersection of each triangle with the plane is a straightforward but important segment of the process. In what follows, we outline the information obtained from this intersection. The three vertices of each triangle are obtained from (67) and the number of sign changes of their w_{ij} coordinates with respect to the plane w = 0 are recorded. Sign changes are identified according to whether each w_{ij} is in $[-1,-\varepsilon),[-\varepsilon,\varepsilon],(\varepsilon,1]$, where ε was defined earlier and denote to be $\varepsilon_{\mathrm{plane}}$. Based on the number of vertices that lie on the plane, ie. in $[-\varepsilon,\varepsilon]$, and the sign changes among the three vertices we distinguish the following cases

1. No intersection (all vertices on one side)
2. Triangle on plane (all vertices on plane, degenerate intersection)
3. Edge intersection (two vertices on plane, third vertex not on plane)
4. Single vertex (one vertex on plane, two vertices on same side of plane)

5. Line intersection with one end on vertex (one vertex on plane, two vertices on either side of the plane)
6. General intersection (no vertices on plane, one vertex on one side of the plane and two on the other side)

Cases 3 through 6 are hereafter collectively called proper intersections. Such information can be used to connect the segments in an unambiguous manner by just comparing integers.

Every such intersection also includes other details of the intersection like identifying indices of vertices and edges at which the intersecting segment ends and the parameter values of the ends of the segments. Specifically, the following fields of information are stored for each such intersection segment

- i, j, k - indices uniquely identifying the triangle
- intflag - type of intersection, one of three cases identified - no intersection, degenerate intersection, proper intersection
- leftent, rightent - type of ending of each intersecting segment - flag indicating if intersecting segment passes through a vertex of the triangle at its left or right end
- leftindu, leftindv - indices indicating the exact vertex or edge at which the segment starts or ends. If the left end of the segment ends at a vertex we use the index ij of the corresponding control point. If it ends on an edge of the triangle, we use the indices of the vertex opposite to the edge on which it ends
- rightindu, rightindv - indices indicating the exact vertex or edge at which the segment starts or ends. Same as the leftindu, leftindv field except that it stores information for the right end
- leftpt, rightpt - the u,v coordinates of the end points of each segment

The above phase gives us a number of disconnected linear segments whose end points are close to the curve. Each of these points may be verified by evaluating the intersection curve at these points to see if an accuracy criterion is met. The exact nature of this criterion is not important for now but only that a certain point satisfies this criterion or not. Each point appears twice as the start and end point of adjoining segments. Since there may be more than one branch in the intersection, these disconnected segments are not, generally, in order and the check for accuracy would be repeated for each point twice. In order to avoid this, we connect all these disconnected segments into one piecewise linear segment before evaluation of each of the points for accuracy. Further, as pointed out earlier, in the initial stages of the iteration it is possible to have spurious loops appearing in the polyhedral intersection which do not form part of the real intersection. This procedure of connecting segments before their evaluation for accuracy also allows us to remove any such spurious intersections by using the property of complete intersection segments within subpatches, ie. that they start and end at distinct borders.

6.2.4 Connection Phase Within a Subpatch

The intersections corresponding to each triangle-plane intersection together form a disconnected set of intersections which possibly contain redundancies and insufficient information because planar facets have been used to approximate a curved surface. The intersection information for **all** the triangles is scanned and triangles having redundant information are marked as having no intersection and triangles with "incomplete information", (adjacent to degenerate triangles that lie completely on the plane) are marked just as those that lie on the plane. The following specific action is taken in the above cases

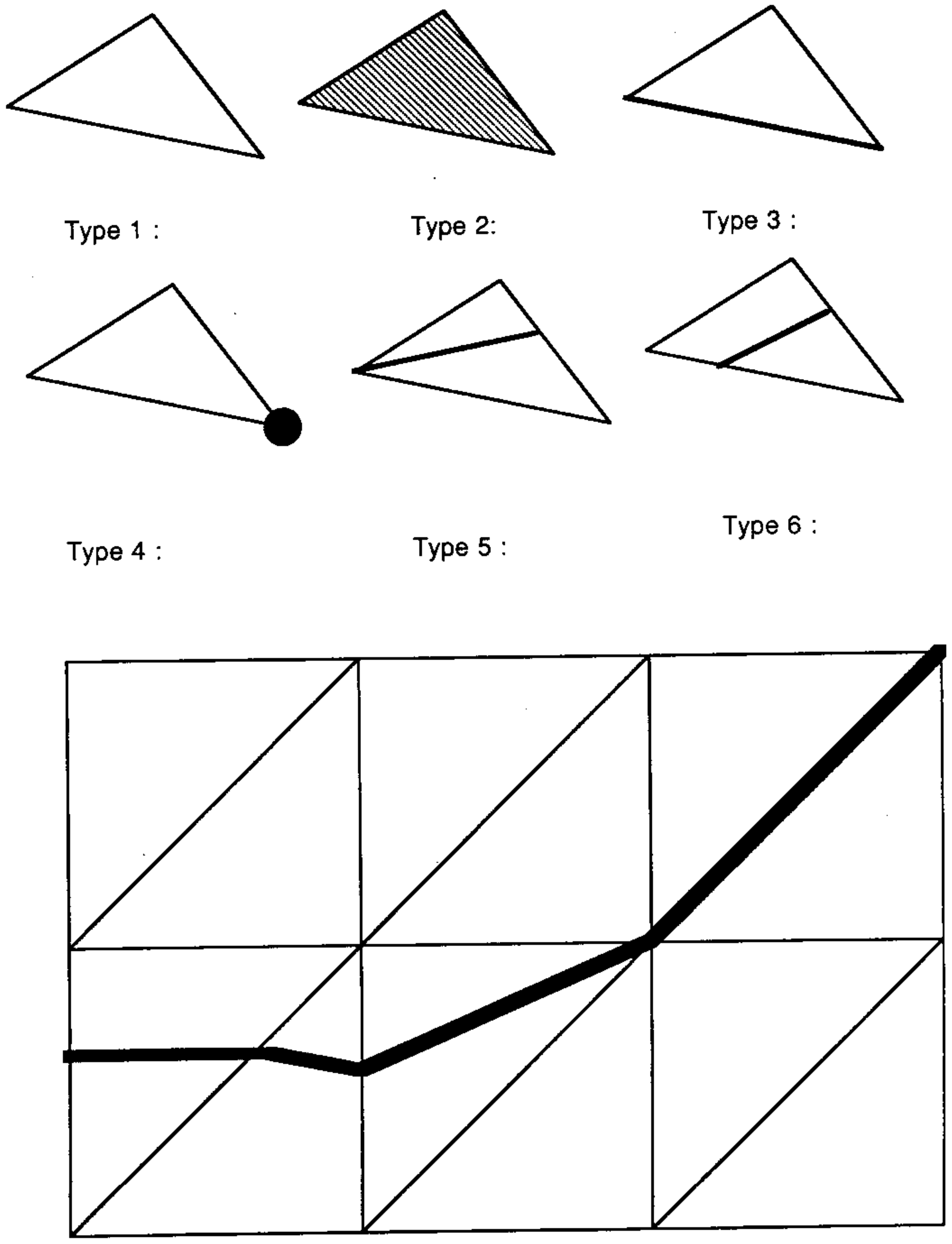
1. Triangles having an edge intersection and in the interior (ie. type 3 and not on a border) have neighboring triangles with identical intersections. One of these triangles is marked as containing no intersection.
2. Triangles with single vertex intersection (ie. type 4, see also Figure 6-2) can occur only where there is a neighboring triangle with interior or edge intersections. The only other non-redundant case of such intersections occurs with isolated points. But these are singularities and can, therefore, occur only at the corners of the patches and not in their interior. Hence, all such triangles may be marked as having no intersection.
3. Triangles that neighbor triangles that are completely on the plane have an edge intersection at the common edge. Such intersections represent "incomplete intersection information" because of the degenerate triangle-plane intersection in the neighboring triangle and lead to incorrect intersections in areas close to singularities where planar facets cannot model a curved surface well. We mark these triangles as also being degenerate to avoid such incorrect connection.

The connection procedure thus depends upon

- the detailed information of the triangle-plane intersection
- the cleaning procedure explained above
- the monotonic form of segments resulting from the way in which the subpatches have been formed, and,
- the property that all branches start and end at borders of the subpatches.

The basic procedure involves searching for starting points of intersection branches along the borders of the subpatch or of triangles with intersections of type 2. Once such an intersecting triangle has been found we proceed to extend this initial intersection through neighboring triangles till it meets another border. The above process is then repeated till no starting intersection can be found. The overall structure of the procedure is best presented in algorithmic

Figure 6-2: Types of triangle-plane intersections



form (also see Figure 6-2)

```
While (A START TRIANGLE CAN BE FOUND)
  {
  start a new list

  make start triangle the present triangle

  deactivate triangle to prevent using it again

  while (NEIGHBOR TO THE PRESENT TRIANGLE CAN BE FOUND)
   {
   Add segment from present triangle to the list

   switch based on type of end point of segment

     VERTEX:

     Find neighbor across vertex at which the present
     segment ended


     EDGE:

     Find neighbor across the edge of
     present triangle at which the present
     segment ended

     }


  Add ending segments where required
  }
```

The major components of the above scheme of connecting the disconnected segments involves the finding of a start triangle and the finding of neighboring triangles across edges and vertices. The starting triangle is found by searching at all corners in sequence and then the borders of the of the patches. For each corner the search is carried out along the border of triangles marked as degenerate if the corner is either a turning or a singular point. A start triangle is found by first looking along such corners and borders till a triangle with a proper intersecting segment is found. Of those, we then select one which has one neighboring triangle with a proper intersection on one side and none on the other side. When such a start triangle has been found, the search for more start triangles is stopped until this segment is completed. If the start triangle was found while searching around a corner with a turning or singular point a small number of collinear segments are added between the appropriate corner and the end of the intersecting segment. This segment is now extended by locating neighbor segments across edges and vertices to form a connected intersection segment. Neighbors across edges are straightforward to find while the neighbors across vertices require the scanning of all neighbors across that vertex to locate the correct intersection. Finally the segments end at a border or a degenerate (type 2) triangle. If it ends at a degenerate triangle, the closest corner is located using the index of the segment end point and the corner indices to unambiguously locate the corner associated with such a triangle.

After completing one such branch, we attempt to find more starting triangles and locate more branches until no starting triangles can be found and we obtain a list of branches each of which is a connected sequence of points. These points are given in the u, v space of the entire patch being intersected.

6.3 The Accuracy of an Approximate Intersection Point

The connected list of points are approximate points that are close to the intersection curve in the parametric space. Their mappings through the parametric surface (9) into three-dimensional space will give points that lie on the parametric surface but do not necessarily lie close enough to the algebraic surface.

Let $\mathbf{P} = (x_i, y_i, z_i, w_i)$ be the point of the parametric patch corresponding to an approximate parametric intersection point Q (u_i, v_i). The value of the algebraic surface equation at point **P** given by $H(x_i, y_i, z_i, w_i)$, where the coefficients of H are normalized as in (5) gives an approximate criterion by which the proximity to the algebraic surface can be judged and, in the limit, as the absolute value of this function becomes smaller than a small positive number, ε, point **P** may be considered to lie on the algebraic surface as well. However the non-dimensional number ε is not, by itself, an indication of the distance between the algebraic surface and the point **P**. However, as pointed out in Appendix II, an estimate of the distance δ between point **P** and the algebraic surface can be computed from $\delta \approx H / |\nabla H|$ where the right hand side is evaluated at point **P** and $|\nabla H|$ is assumed to be non-zero, i.e. **P** is not close to singularities of the algebraic surface $H = 0$. Distance considerations are of special importance in solid modeling applications and affect the reliability of the design database. The algorithm developed in this work employs the above criterion $(\delta \leq \varepsilon_{3D})$ to judge the accuracy of points in three-dimensional space.

6.4 Iteration

If every point Q from the list of branches in the domain of each subpatch is considered acceptable according to criteria established in the Section 6.3, the iteration is complete and the points provide a piecewise linear approximation of the intersection in the parametric space of the patch. If a point Q does not satisfy the accuracy requirement, then its coordinates u and v are used to insert a knot in the u and v knot vectors, respectively, of the surface (60) which generated these intersections by faceting. This procedure is carried out for all points. In order to avoid introducing multiplicities at a certain value of u or v because two distinct failing points have a common u or v, we first obtain a list of the failing points, separate the (u,v) pairs into two sorted lists of u and v values and remove any duplications in the u or v to given tolerance ε_{sort}. Once the new knot vectors are created, a subdivision procedure based on the Oslo algorithm is invoked. This forms a computationally intensive part of the iteration step. It allows the simultaneous insertion of an arbitrary number of non-uniformly spaced knots. Non-uniform insertion of knots allows us to introduce a finer approximation at regions where the curve is not well approximated, providing the basis for an **adaptive** algorithm. Algorithmically stated, we do the following

```
If (All accuracy criteria are met for a given point)

     point is good
else
     point is bad


If ( all points in all branches are good )

     The curve branches in this subpatch
     are good
else
     {
     insert knots at failed (u, v)
     points in the subpatch

     use Oslo algorithm to get the
     finer control polyhedron
     }
```

The finer control polyhedron will now be closer to the surface and, hence, the repetition of the intersection of triangles, connection and testing for accuracy will yield a new intersection approximation that is better than that obtained in the preceding stage. This process of iteration may be continued till all points in all intersection branches of the subpatch are of sufficient accuracy. Convergence is guaranteed because, in the limit, the control polyhedron tends to the surface [31, 13].

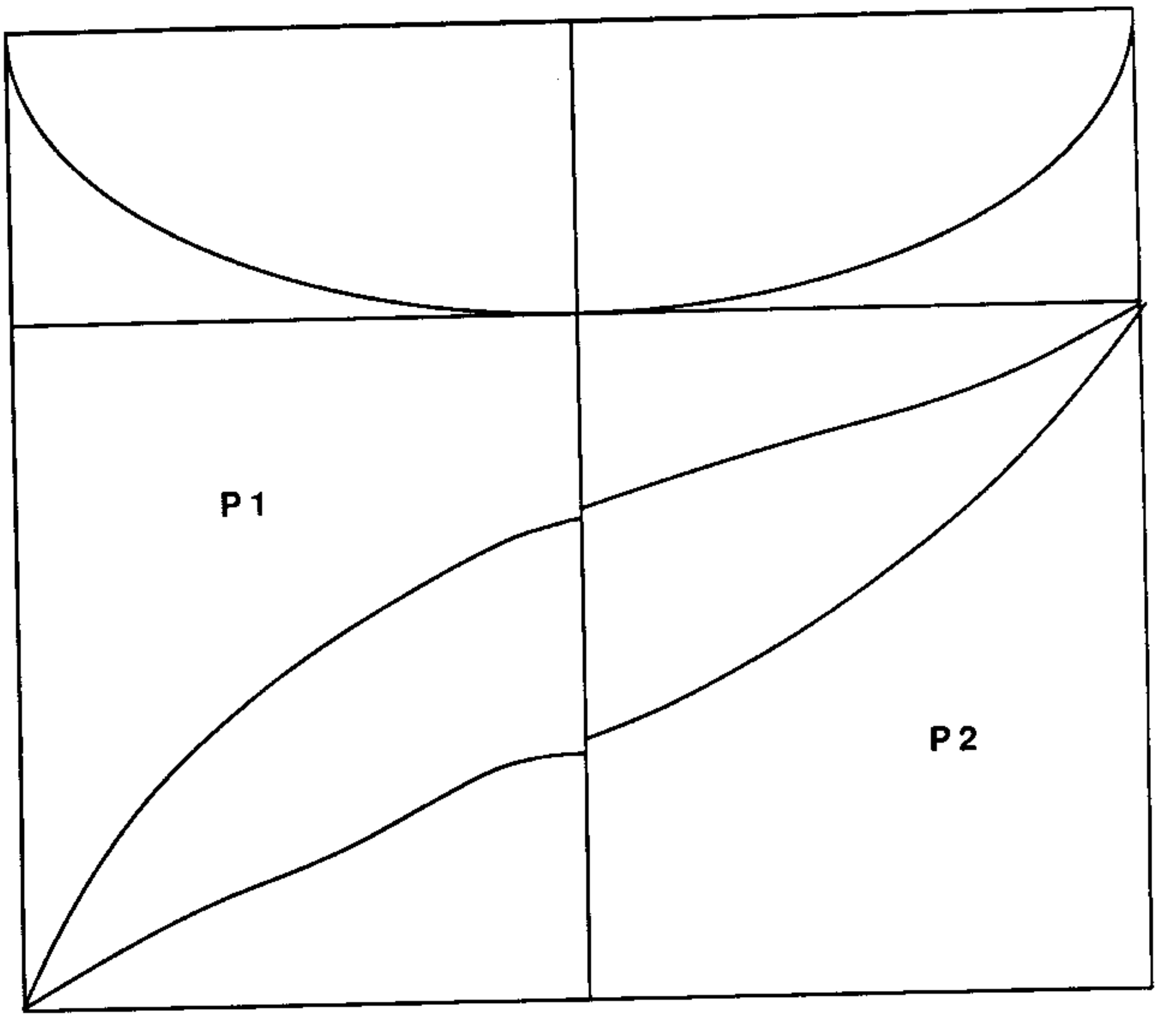
6.5 Connection Phase across Subpatches

The above tracing procedure is repeated for each one of the subpatches (63) generated by splitting of the surface at all significant points. It is possible (and, in fact, frequent) that such partition produces splitting of segments at regular points at which no special reasons exist for discontinuity. This is a by-product of the particular splitting procedure employed (see Figure 6-3). In this Section, we present a method of correcting this artificial discontinuity, present when a finite amount of subpatch subdivision is carried out, and combining the solutions from each of the subpatches into a complete intersection curve. Once this procedure is completed, a list of separate segments of the curve starting and ending at all significant points is produced. The tracing procedure used above does not produce lists which contain isolated points as solution points. In order to rectify the final solution we add one-point segments corresponding to all isolated points. These can be determined by comparing the ends of all segments with the list of significant points. During the connection phase within subpatches, indices specifying the exact border of each subpatch in which a segment ends are recorded in the data structure for each segment. This expedites the connection across subpatches outlined below.

Let us denote two subpatches, which have common borders with such an artificial discontinuity, to be P1 and P2 (see Figure 6-3). Usually P1 and P2 each have one segment reaching the common border. In general these two do not meet at a common point on the border. This occurs because the two subpatches could have been subdivided to differing levels of refinement and correspondingly the planar facet intersections between the triangles and the plane on either side of the common border need not produce the same intersecting segment end points. The exact point of the intersection may be found by solving for the intersection of the algebraic curve with the common border of the two neighboring subdomains corresponding to subpatches P1 and P2. This point can then replace the two mismatched end points. The two segments can now be connected to form a single larger connected segment. A similar approach is used for other segments and across other patches.

In the event that more than one segment in each of P1 and P2 have end points on the common border a tolerance based criterion needs to be employed to identify the matching pairs of segments. In the unlikely event of ambiguity due to the finite tolerance employed and

Figure 6-3: Connecting Segments across Subpatches



computation inaccuracies, it becomes necessary to evaluate the derivatives at each of the points to determine the matching pairs and connect them as explained above. If this does not provide unique matching pairs, then higher order derivatives have to be evaluated until they unambiguously establish matching pairs. An important point to note here is that such border points and the matching of the corresponding segments across the border cannot involve a singularity such as self-intersection because all singularities have been found a priori and are located at corners of the subpatches.

At the conclusion of the above process, a list of separate segments of the curve of intersection starting and ending at all significant points within or on the border of the parametric domain of the polynomial patch becomes available. The segments ending at turning points may be connected across such points without difficulty by simply matching position and without the need for any tolerancing since these points have been computed apriori. In the case of segments ending at singular points the desingularization method studied by Bajaj et al [8] may be employed to connect the proper intersection branches across such points.

6.6 Tracing Other Polynomial Patches of the B-Spline Surface

As mentioned in Section 4.2, we trace the intersection curve between the B-spline patch and the algebraic surface by splitting the B-spline patch into its constituent polynomial patches and tracing each one of them separately. The above sections describe the tracing of the intersection of one such polynomial patch. Similar intersection problems are solved for all other patches to get similar solutions of connected lists of points that all end at significant points. The solutions may be combined into larger segments across border points by simply matching end points between the segments of neighboring polynomial patches. Finally, the points in each one of the segments is mapped into three-dimensional space to obtain the actual intersection curve. The points may also be retained in the parameter space for some applications, such as processing of trimmed patches in a solid modeling environment [19].

7. APPLICATIONS

7.1 Preliminary Remarks

The method outlined in the earlier sections has been implemented and tested to verify its reliability and to evaluate the accuracy achieved in the various stages of computation. Our implementation is written in C on a DEC Vax Station II GPX machine operating under the Unix system and employing the Silicon Graphics Remote Graphics Library to drive a Silicon Graphics IRIS 3030 workstation.

The method was implemented and tested in two stages

1. The computation of the representation of the intersection curve in the parametric space of the patch from given intersecting surfaces and the computation of its significant points
2. The tracing of the intersection curve given its representation in the Bernstein basis and its significant points ie. the process of obtaining various branches that are linear spline approximations of the curve which run from significant point to significant point.

This separation was used so as to be able to evaluate each element of the method independently. In our current implementation, for example, the tracing portion is not limited by the degree of the curve, so that curves of arbitrary degree can, in principle, be traced. On the other hand, the computation of significant points, at present limited to low degree curves, requires consideration of a possible significant loss in accuracy with increased degree.

7.2 Tracing Known Curves with Varied Singularities

Extensive examples of planar algebraic curves can be found in Walker [49] and Lawrence [33] and more recently in Geisow [21] who used them to test methods to trace algebraic curves. A number of examples drawn from these references have been traced using the method described above. The standard form of these curves given in the literature is the monomial form with integer coefficients. The characteristic polynomials of these curves were derived using elimination techniques in rational arithmetic. In most instances exact solutions of these equations in terms of rational numbers or numbers involving radicals of rationals are possible and, hence, such solutions were obtained with 16 decimal digit accuracy. The remaining real solutions within the window of interest were obtained using Jenkins's algorithm [28] implemented within MACSYMA [36] to the best possible accuracy employing 16 decimal digit floating point arithmetic. The range of curves chosen involve a maximum of degree six in each of the two variables and exhibit double, triple or quadruple singular points. For double points, special cases involving an isolated point (hermit point or acnode); node (crunode or self-intersection) and cusp (spinode) have been studied. Cusps of the first and second kind have been studied, i.e. when curve arcs lie on either or one side of the double tangent. Finally, double cusps (tacnodes or points of osculation) at which the two arcs extend in both directions of tangents have been tested. Some of these examples have a number of turning and border points depending on the window of interest. The tolerances introduced in Sections 5 and 6 were held constant and equal to the following values

- $\varepsilon_{sort} = 1.0 \times 10^{-6}$, used to sort significant points
- $\varepsilon_{plane} = 1.0 \times 10^{-12}$, used to evaluate sign changes of weights w_{ij}
- $\varepsilon_{fuv} = 1.0 \times 10^{-3}$, used to check the value of the normalized algebraic curve equation

Note that the weights w_{ij} in the Bernstein basis have been normalized so that $|w_{ij}| \leq 1$ and, therefore, the surface $w = F(u,v)$ lies in $[-1,1]$ within the window of interest. When the equation of the intersection curve only is given and there is no generating three-dimensional intersection problem, the accuracy criterion of Section 6.3 cannot be implemented as stated. However, when $|F(u,v)| \leq \varepsilon << 1$ is obtained, a distance estimate from an approximate point u, v close to the curve to the curve $F = 0$ can be approximated by $\delta = |F|/|\nabla F|$ evaluated at the approximate point, valid when $|\nabla F| \neq 0$. The accuracy criterion used to trace the algebraic curve examples of this Section is $\delta \leq \varepsilon_{2D}$ where $\varepsilon = 1.0\text{x}10^{-3}$.

Each of the typical examples chosen are presented in the following format

- The monomial representation of the curve and the window of interest
- The significant points of the curve
- A picture of the actual trace of the curve

The following is a list of the examples that we have included in this report

1. Double point - (acnode, isolated point [49])
2. Double point - (crunode, Tschirnhausen's cubic [49])
3. Double point - (crunode, folium of Descartes [49])
4. Double point - (crunode, torus-plane intersection at a saddle point [38])
5. Double point - (cusp of the first kind, Isochrone [33])
6. Double point - (cusp of the first kind, Cardioid [33])

7. Double point - (cusp of the second kind, Ramphoid [33])
8. Double point - (Double cusp ie. tacnode, Hippopede [33])
9. Double point - (Geisow's tacnode and crunode [21])
10. Double point - (Geisow's multiple crunode case [21])
11. Triple point [49]
12. Quadruple point [49]
13. Reducible curve - (crunode)

7.3 Intersections of Algebraic Surfaces with Rational Biquadratic and Bicubic Patches

The first part is tested by the study of some standard surfaces. Our present implementation of this part handles intersections of planes with rational biquadratic and bicubic B-spline patches. The input consists of the position and orientation of the algebraic surface and the patch in a world coordinate system along with the representation of each of the surfaces in their local coordinate systems. The plane is represented by the normal to the plane and a point on the plane given in the world coordinate system. The rational B-spline patch is represented by the definition of a local system in world space and the location of the control points in the local system. The tolerances introduced in this report were held constant and equal to the following values

- $\varepsilon_{sort} = 1.0 \times 10^{-6}$, used to sort significant points.
- $\varepsilon_{plane} = 1.0 \times 10^{-12}$, used to evaluate sign changes of weights w_{ij}
- $\varepsilon_{fuv} = 1.0 \times 10^{-3}$, used to check the value of the normalized algebraic curve equation.
- $\varepsilon_h = 1.0 \mathrm{x} 10^{-3}$, used to check the value of the normalized algebraic surface equation
- $\varepsilon_{3D} = 1.0 \mathrm{x} 10^{-3}$, used to check the distance accuracy of an approximate intersection point

We have chosen the following examples of rational polynomial parametric surface patches to illustrate the handling of various features of the curve of intersection by our algorithm

1. A simple biquadratic Bezier patch and a plane leading to an algebraic curve with four turning points
2. A torus represented as a rational biquadratic B-spline patch intersected with a plane tangent at an inner saddle point of the torus
3. A torus represented as above with a nearly tangent plane leading to a small isolated loop

$F(u,v) = u^3 + u^2 + v^2 = 0$

Window of interest : [-2.0 1.0] x [-1.0 1.0]

Significant Points :

Border points	-1.465571231876771	1.000000000000000
	-1.465571231876771	-1.000000000000000
Turning points	-1.000000000000000	0.000000000000000
Singular points	0.000000000000000	0.000000000000000

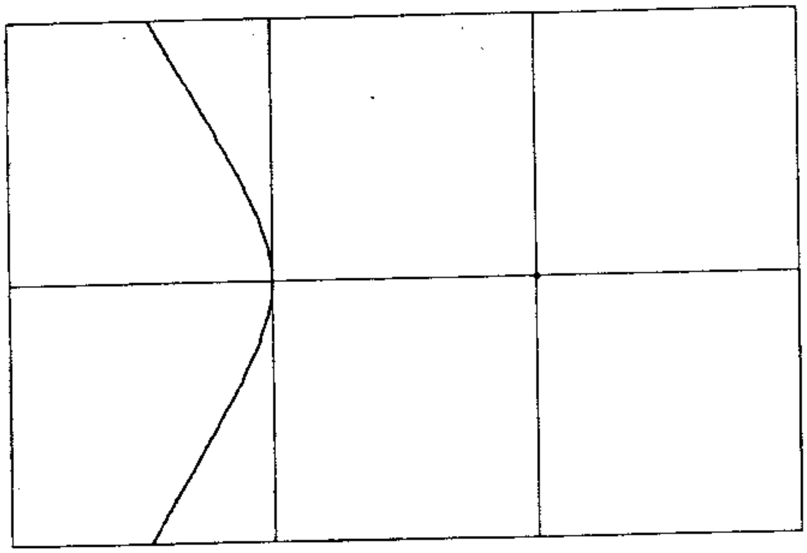


Figure 7-1: Double point - (acnode, isolated point)

$F(u,v) = 15v^2 - 5u^2 - u^3 = 0$

Window of interest : [-5.0 2.0] x [-2.0 2.0]

Significant Points :

Border points	2.000000000000000	1.366260102127946
	2.000000000000000	-1.366260102127946
Turning points	-5.000000000000000	0.000000000000000
	-3.333333333333333	1.111111111111111
	-3.333333333333333	-1.111111111111111
Singular points	0.000000000000000	0.000000000000000

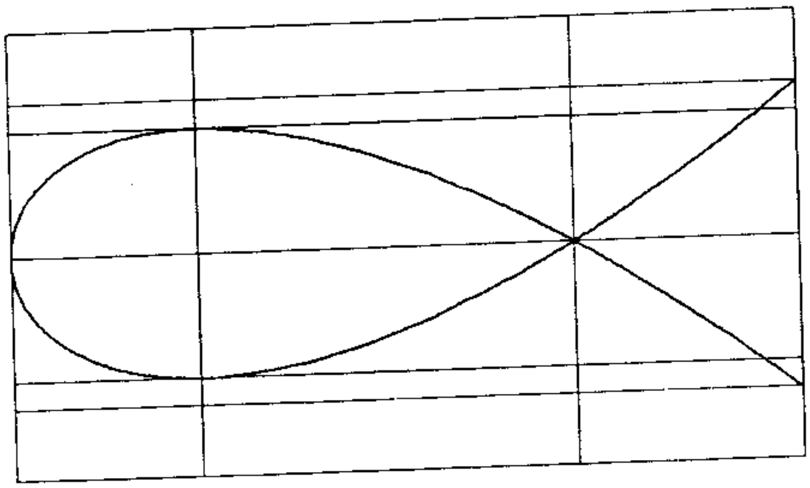


Figure 7-2: Double point - (crunode, Tschirnhausen's cubic)

$F(u,v) = u^3 - 3uv + v^3 = 0$

Window of interest : [-3.0 2.0]x[-2.0 2.0]

Significant Points :

Border points	1.107147564435333	-2.0000000000000000
	-2.951373035591441	2.0000000000000000
Turning points	1.259921049894873	1.5874010519681996
	1.587401051968196	1.2599210498948731
Singular points	0.000000000000000	0.000000000000000

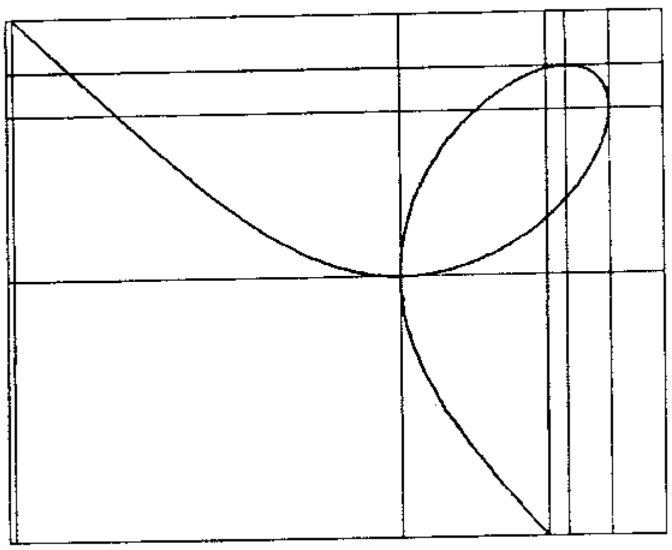


Figure 7-3: Double point - (crunode, folium of Descartes)

$F(u,v) = u^4 - 7200u^2 + 2u^2v^2 + 7200v^2 + v^4 = 0$

Window of interest : [-90.0 90.0] x [-30.0 30.0]

Significant Points :

Turning points	-84.8528137423857	0.00000000000000
	-51.9615242270663	30.00000000000000
	51.9615242270663	-30.00000000000000
	84.8528137423857	0.00000000000000
	51.9615242270663	30.00000000000000
	-51.9615242270663	-30.00000000000000
Singular points	0.0000000000000	0.00000000000000

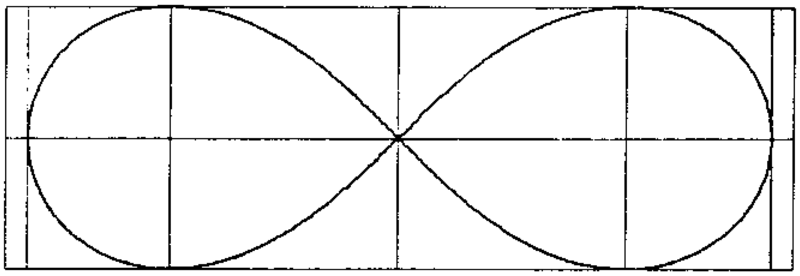


Figure 7-4: Double point - (crunode, torus plane intersection at a saddle point)

$F(u,v) = v^2 - u^3 = 0$

Window of interest : [-1.0 1.0] x [-1.1 1.1]

Significant Points :

Border points	1.000000000000000	1.000000000000000
	1.000000000000000	-1.000000000000000
Singular points	0.000000000000000	0.000000000000000

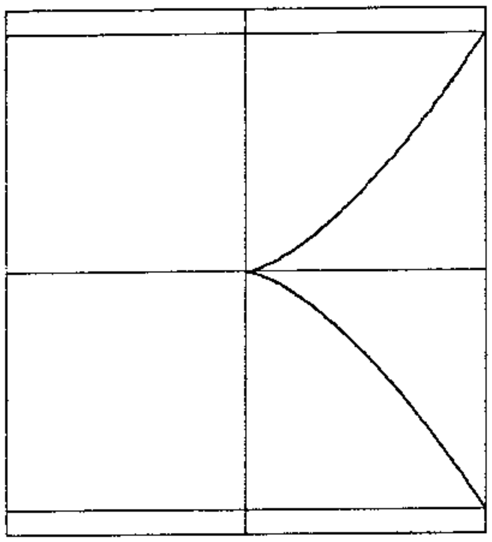


Figure 7-5: Double point - (cusp of the first kind, Isochrone)

$F(u,v) = u^4 - 4u^3 + 2u^2v^2 - 4uv^2 - 4v^2 + v^4 = 0$

Window of interest : [-0.5 4.0] x [-3.0 3.0]

Significant Points :

Turning points	1.500000000000000	2.598076211353316
	1.500000000000000	-2.598076211353316
	4.000000000000000	0.000000000000000
	-0.500000000000000	-0.866025403784438
	-0.500000000000000	0.866025403784438
Singular points	0.000000000000000	0.000000000000000

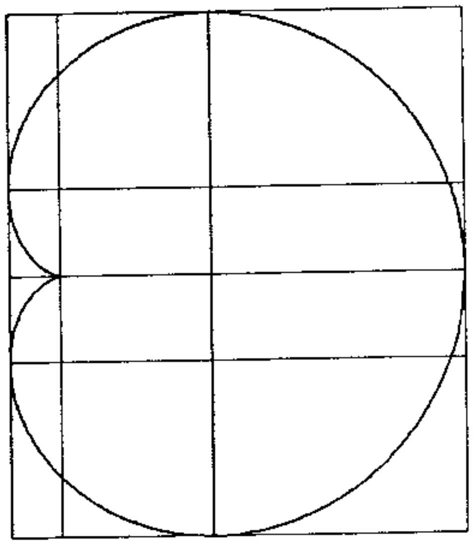


Figure 7-6: Double point - (cusp of the first kind, Cardioid)

$F(u,v) = u^4 - 2u^2v - uv^2 + v^2 = 0$

Window of interest : [-2.0 2.0]x[-2.0 2.0]

Significant Points :

Turning points	0.923216214762122 1.000000000000000	1.161602642611495 1.000000000000000
Singular points	0.000000000000000	0.000000000000000

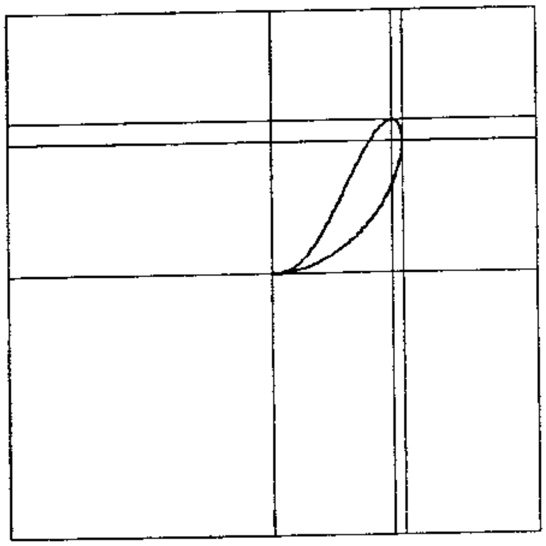


Figure 7-7: Double point - (cusp of the second kind, Ramphoid)

$F(u,v) = u^4 - 4u^2 + 2u^2v^2 + v^4 = 0$

Window of interest : [-2.0 2.0] x [-2.0 2.0]

Significant Points :

Turning points	1.0000000000000000	1.000000000000000
	-1.0000000000000000	1.000000000000000
	1.0000000000000000	-1.000000000000000
	-1.0000000000000000	-1.000000000000000
	2.0000000000000000	0.000000000000000
	-2.0000000000000000	0.000000000000000
Singular points	0.000000000000000	0.000000000000000

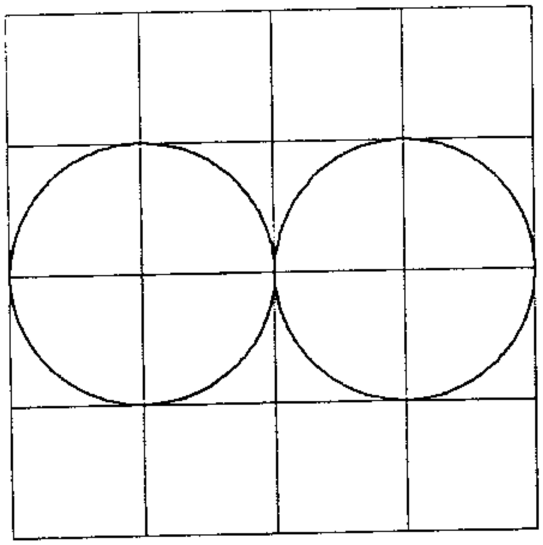


Figure 7-8: Double point - (Double cusp ie. tacnode, Hippopede)

$F(u,v) = 2u^4 - 3u^2v + v^2 - 2v^3 + v^4 = 0$

Window of interest : [-2.0 2.0] x [-1.0 3.0]

Significant Points :

Turning points	-1.243179443537765	2.060660171779821
	1.243179443537765	2.060660171779821
	0.236555716204104	0.300238544000019
	-0.236555716204104	0.300238544000019
	1.496920322406107	1.758935817927207
	-1.496920322406107	1.758935817927207
Singular points	0.000000000000000	0.000000000000000
	0.000000000000000	1.000000000000000

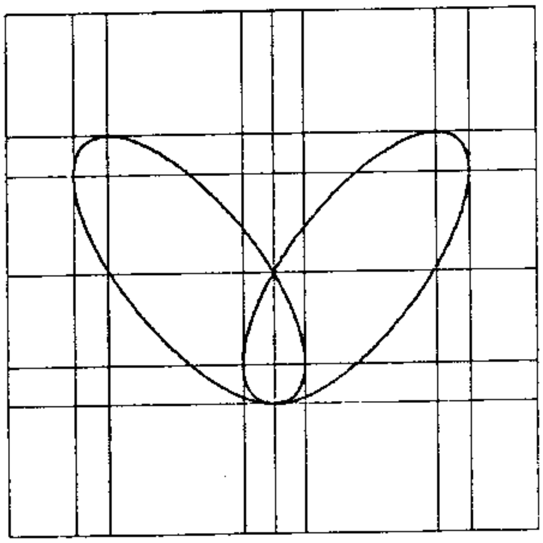


Figure 7-9: Double point - (Geisow's tacnode and crunode)

$F(u,v) = -6u^4 + 21u^3 - 19u^2 - 6u^2v^2 + 11uv^2 + 3v^2 - 4v^4 = 0$

Window of interest : [-0.5 2.0] x [-1.5 1.5]

Significant Points :

Turning points	0.500000000000000	-1.118033988749895
	0.500000000000000	1.118033988749895
	1.206854609343685	1.032481733179492
	1.206854609343685	-1.032481733179492
	1.443145390656315	0.818676658181160
	1.443145390656315	-0.818676658181160
	1.500000000000000	-0.866025403784438
	1.500000000000000	0.866025403784438
	-0.100000000000000	-0.479583152331272
	-0.100000000000000	0.479583152331272
Singular points	0.000000000000000	0.000000000000000
	1.000000000000000	1.000000000000000
	1.000000000000000	-1.000000000000000

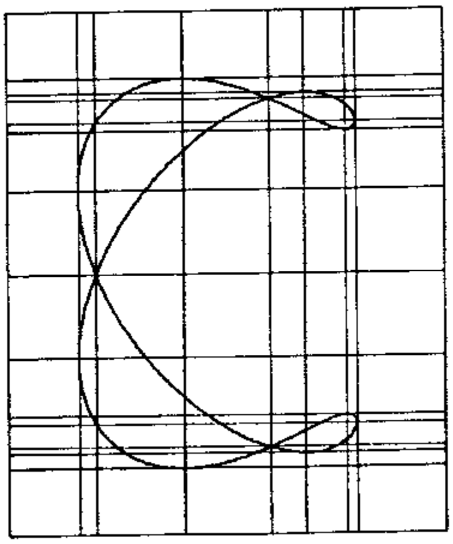


Figure 7-10: Double point - (Geisow's multiple crunode case)

$F(u,v) = u^4 + 3u^2v + 2u^2v^2 - v^3 + v^4 = 0$

Window of interest : [-2.0 2.0] x [-2.0 2.0]

Significant Points :

Turning points	0.000000000000000	1.0000000000000000
	-0.726184377413891	-0.5625000000000000
	0.726184377413891	-0.5625000000000000
	-0.880086296523043	-0.4448027481129402
	0.880086296523043	-0.4448027481129402
	-0.184504364914095	0.6323027481129402
	0.184504364914095	0.6323027481129402
Singular points	0.000000000000000	0.000000000000000

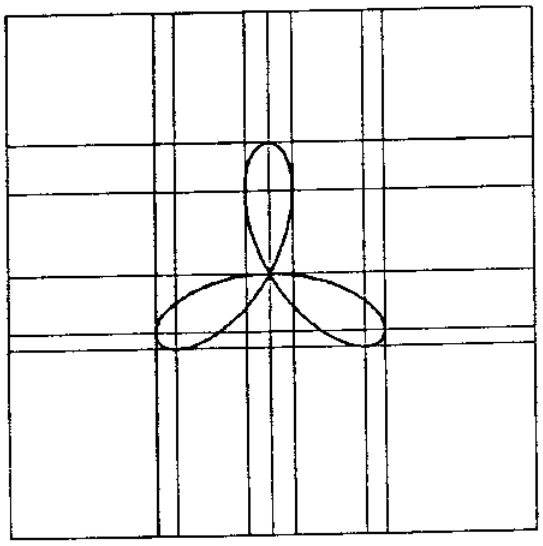


Figure 7-11: Triple point

$F(u,v) = u^6 + 3u^4v^2 - 4u^2v^2 + 3u^2v^4 + v^6 = 0$

Window of interest : [-1.0 1.0] x [-1.0 1.0]

Significant Points :

Turning points	-0.544331053951817	-0.769800358919501
	-0.544331053951817	0.769800358919501
	0.544331053951817	-0.769800358919501
	0.544331053951817	0.769800358919501
	-0.769800358919501	-0.544331053951817
	-0.769800358919501	0.544331053951817
	0.769800358919501	-0.544331053951817
	0.769800358919501	0.544331053951817
Singular points	0.000000000000000	0.000000000000000

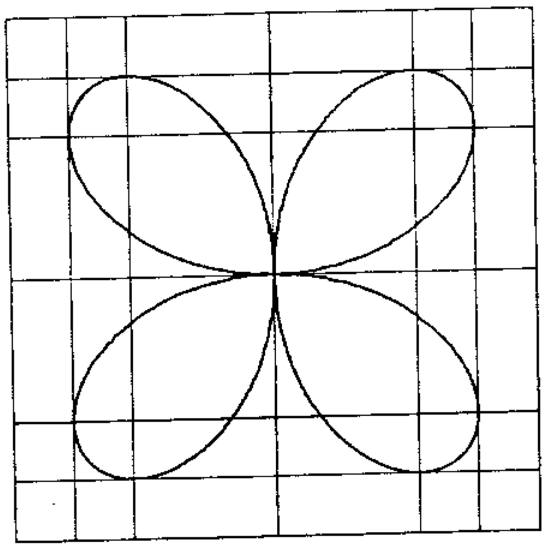


Figure 7-12: Quadruple point

$F(u,v) = (u - v)(u^2 + v^2 - 1) = 0$

Window of interest : [-1.0 1.0] x [-1.0 1.0]

Significant Points :

Turning points	0.0000000000000000	1.0000000000000000
	-1.0000000000000000	0.0000000000000000
	0.0000000000000000	-1.0000000000000000
	1.0000000000000000	0.0000000000000000
Singular points	0.7071067811865475	0.7071067811865475
	-0.7071067811865475	-0.7071067811865475

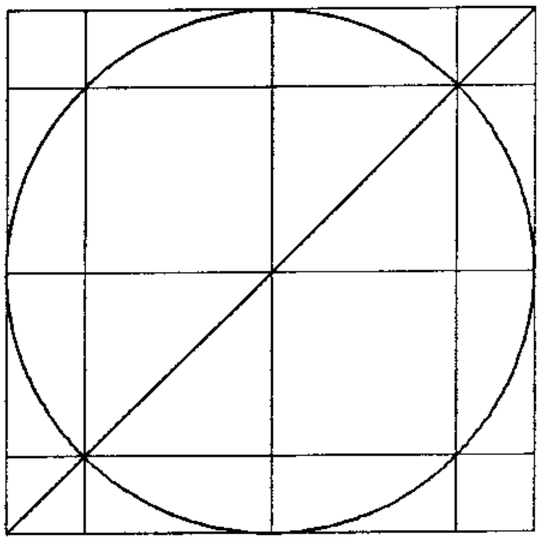


Figure 7-13: Reducible curve - (crunode)

4. A torus represented as above with a plane tangent to it at two points on either side of the equatorial plane
5. A bicubic patch with a plane leading to an algebraic curve with one turning point

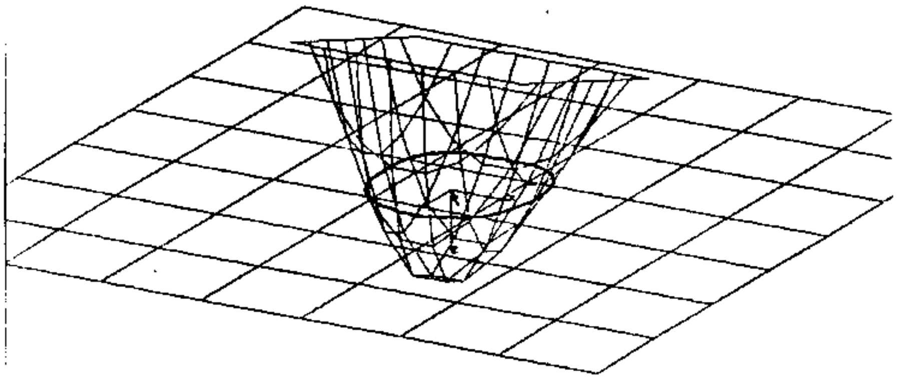


Figure 7-14: Biquadratic-plane, four turning points

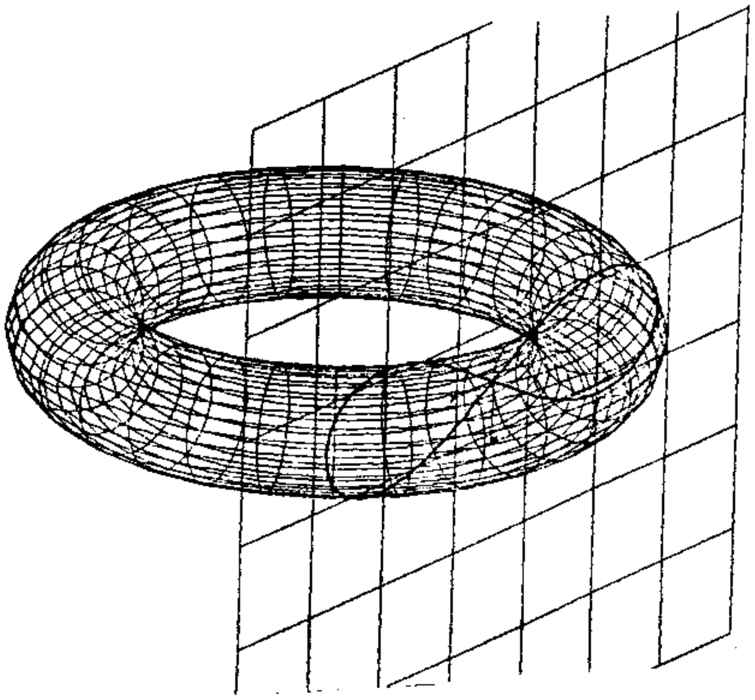


Figure 7-15: Torus-plane, inner saddle point tangency

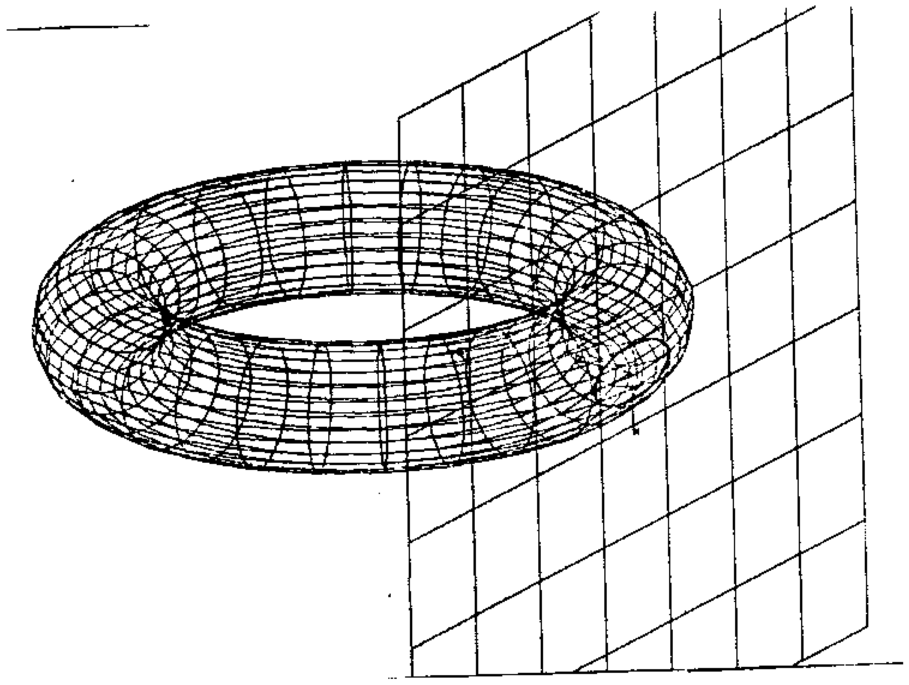


Figure 7-16: Torus-plane, small isolated loop

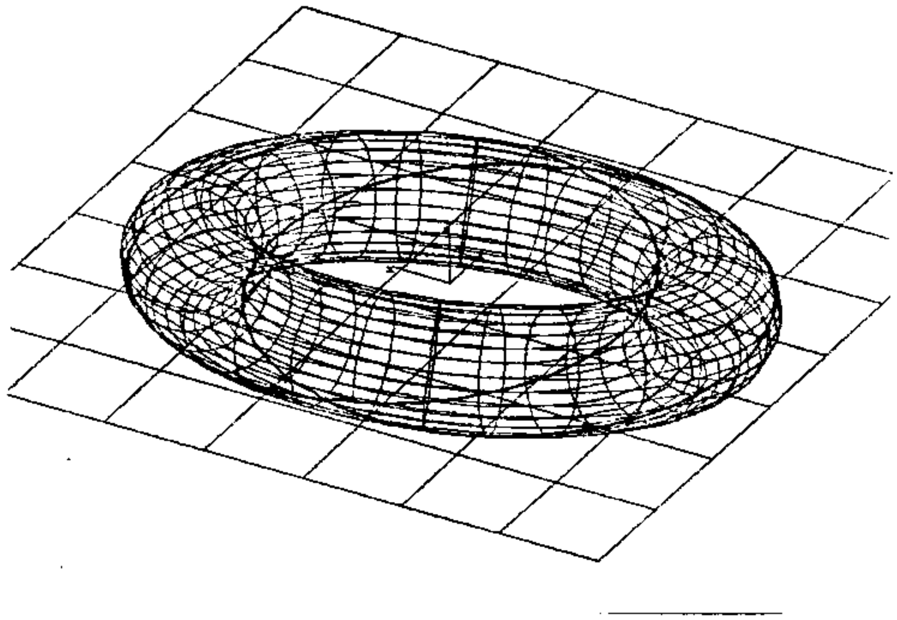


Figure 7-17: Torus-plane, double tangency

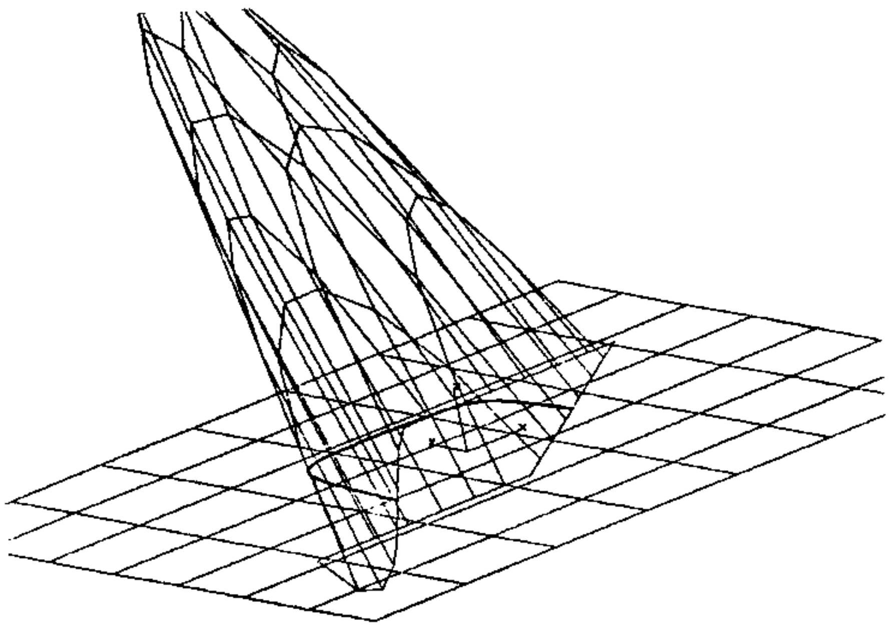


Figure 7-18: Bicubic-plane, one turning point

8. CONCLUSIONS

This report presents the basic components of a new algorithm to trace a planar algebraic curve within a rectangular parallelogram arising in the context of automatic interrogation of intersections of algebraic surfaces and piecewise continuous rational polynomial parametric surface patches. The method combines the advantageous features of analytic representation of the governing equation in the Bernstein basis with adaptive subdivision techniques and the a priori computation of border, turning and singular points to provide the basis for a reliable and efficient solution procedure.

The analytic representation of the curve provides the capability to transform the problem at hand to the intersection of an auxiliary parametric surface patch and an auxiliary plane and is a key feature of our method. Conversion of the representation of the auxiliary parametric patch to the Bernstein-Bezier form allows the use of subdivision and faceting techniques to obtain an approximation of the curve of intersection in the parametric space. Further, due to the variation diminishing property of this basis, subdivision and faceting prevents small loops from being missed. An additional key element of our method is the computation of the significant points of the curve allowing subdivision and faceting techniques to provide approximations of the curve with correct connectivity. The process of determining significant points, at present, involves the solution of univariate polynomials whose coefficients are obtained by expansion of determinants. It has been observed that the accuracy and efficiency with which the determination and solution of such equations can be obtained may be below the requirements of solid modeling applications and, hence, the above method of computing significant points (particularly turning and singular points) is limited to low degree intersections. A preliminary investigation into the use of minimization methods to obtain these significant points, however, indicated that such methods show promise as they do not involve expansion of determinants and the solution of high degree polynomials. The application of minimization techniques to compute turning and singular points requires initial approximations for all such points which can be obtained, using subdivision and faceting techniques, because the variation diminishing property is valid for each parametric line of the control surface of the transformed problem. Detailed investigation of these ideas is recommended. Tracing of the curve with the help of the computed significant points is relatively independent of problems of high degree and performs well in test examples from the literature.

Finally we point out the parallel character of the above minimization method and tracing procedure at the stage of converting polynomial patches into smaller subpatches each of which may be traced independently and intersecting faceting approximations of subpatches with a plane. This advantageous feature of our method is important for the computation of intersections in large-scale real-time applications.

I. Basis Transformations

In this Appendix, we develop the relations needed to transform polynomials expressed in the monomial (power) basis to the Bernstein basis and vice-versa.

Let P be a polynomial expressed in the monomial basis

$$P(u) = \sum_{i=0}^{n} a_i u^i = [\,A\,][\,M_n(u)\,]^T \tag{68}$$

where [A] is a row vector of coefficients a_i, $i = 0,1,...n$ and where $[M_n(u)]$ is a row vector of monomials i.e. u^i, $i = 0,1,...n$. Similarly, let the above polynomial P be expressed in the Bernstein basis as

$$Q(u) = \sum_{i=0}^{n} c_i B_{i,n}(u) = [\,C\,][\,B_n(u)\,]^T \tag{69}$$

where [C] is the row vector of coefficients, c_i, $i = 0,1,...n$ and $[B_n(u)]$ is the row vector of Bernstein polynomials, $B_{i,n}(u)$, $i = 0,1...n$,

$$B_{i,n}(u) = \frac{n!}{i!\,(n-i)!}\, u^i (1-u)^{n-i} \qquad i = 0,1,...n \tag{70}$$

Since the power and Bernstein bases both span the space of polynomials of given degree n, each power basis function may be expressed as a linear combination of the n+1 Bernstein basis functions [20, 16]

$$[\,M_n(u)\,] = [\,B_n(u)\,][\,T^{mb}(n)\,] \tag{71}$$

where $[T^{mb}(n)]$ is a square matrix of size (n+1) given by

$$\begin{aligned} T^{mb}_{ij}(n) &= \binom{j}{i}\binom{n}{i}^{-1} && for\ i \le j \\ &= 0 && for\ \ i > j \end{aligned} \tag{72}$$

Vice-versa, each Bernstein basis function may be expressed as a linear combination of the (n+1) power basis functions

$$[\,B_n(u)\,] = [\,M_n(u)\,][\,T^{bm}(n)\,] \tag{73}$$

where $[T^{bm}(n)]$ is a square matrix of size (n+1) given by

$$\begin{aligned} T_{ij}^{bm}(n) &= (-1)^{j-i}\binom{n}{j}\binom{j}{i} \qquad && for\ i \le j \\ &= 0 && for\ i > j \end{aligned} \tag{74}$$

The transformation from the monomial to the Bernstein basis may be specified by

$$[\,C\,] = [\,A\,][\,T^{mb}(n)\,]^T \tag{75}$$

and from Bernstein to the monomial basis

$$[\,A\,] = [\,C\,][\,T^{bm}(n)\,]^T \tag{76}$$

From (75) and (76) it is obvious that

$$[\,T^{mb}(n)\,] = [\,T^{bm}(n)\,]^{-1} \tag{77}$$

The elements of the two transformation matrices are rational numbers and are only a function of the degree of the polynomial involved. Hence it is possible to compute these numbers using just rational arithmetic and convert them to floating point arithmetic by means of a single division preserving the full precision of the machine. This may be important if a high degree of accuracy on the transformed coefficients is required.

We now apply the above results to convert an algebraic curve expressed in the monomial basis to the Bernstein basis. Let the algebraic curve be specified by

$$F(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} a_{ij}\, u^i v^j = [\,U\,][\,A\,][\,V\,]^T \tag{78}$$

where U is a row vector of elements u^i, $i = 0,1,...m$ and V is a row vector of elements v^i, $i = 0,1,...n$ and and A is matrix of size (m+1) x (n+1) with elements a_{ij}.

Since U and V are the row vectors corresponding to M in (68) with variables u and v, relation (71) gives

$$\begin{aligned} [\,U\,] &= [\,B_m(u)\,][\,T^{mb}(m)\,] \\ [\,V\,] &= [\,B_n(v)\,][\,T^{mb}(n)\,] \end{aligned} \tag{79}$$

where $[B_m(u)]$ and $[B_n(v)]$ are vectors of size m+1 and n+1 respectively and $[T^{mb}(m)]$ and $[T^{mb}(n)]$ are square matrices of size m+1 and n+1 respectively.

Hence (78) becomes

$$F = [\,B_m(u)\,][\,T^{mb}(m)\,][\,A\,][\,T^{mb}(n)\,]^T[\,B_n(v)\,]^T \tag{80}$$

and the matrix [W] of Bernstein coefficients, w_{ij}, is given by

$$[\,W\,] = [\,T^{mb}(m)\,][\,A\,][\,T^{mb}(n)\,]^T \tag{81}$$

and can be computed as a product of three matrices.

II. Distance of a Point from an Implicit Surface

In this Appendix, we derive an approximate expression for the shortest distance from any point close to an implicit surface to the surface itself.

Let **P** be a point known to be close to an implicit surface F(**R**) = 0. Let **Q** be some point on the surface, F, close to point **P**. Then the following Taylor expansion may be employed

$$F(\mathbf{Q}) = F(\mathbf{P}) + t(aF_x + bF_y + cF_z) + O(t^2) = 0 \tag{82}$$

where a, b and c are the direction cosines of a straight line that passes through **P** and **Q**; the derivatives are evaluated at the point **P** and $|\nabla F| \neq 0$ in a neighbourhood of **P** where ∇ denotes gradient and t is the parameter describing the straight line. The direction cosines satisfy the relation

$$a^2 + b^2 + c^2 - 1 = 0 \tag{83}$$

The absolute value of the parameter t also represents the Euclidean distance between points **P** and **Q** when a, b and c satisfy the above relationship. Solving (82) for t and neglecting $O(t^2)$ terms we get

$$|t| \approx \frac{|F|}{|aF_x + bF_y + cF_z|} \tag{84}$$

In order to minimize this distance $|t|$, we need to maximize the denominator of (84) i.e. we may

$$\mathit{Maximize}\ \ [aF_x + bF_y + cF_z]^2 \tag{85}$$

subject to contraint (83). This problem can be solved using a direct method or the Lagrange multiplier technique [26] to give the following solutions for a, b and c

$$[a, b, c] \approx \pm \frac{\nabla F}{|\nabla F|} \tag{86}$$

Substituting (86) in (84) we get

$$|t|_{min} \approx \frac{|F|}{|\nabla F|} \tag{87}$$

Hence, the shortest distance from any point close to an implicit surface to the implicit surface itself may be approximately computed using (87). The above expression is obviously exact when F represents a plane and was found to give good results for higher degree algebraic surfaces.

III. Classification of Singular Points

In this Appendix, we define and classify the singular points of an algebraic curve.

A point P on an algebraic curve F(u,v) = 0 is called a singular point if the partial derivatives F_u and F_v vanish at P. Singular points are classified according to the number of partial derivatives vanishing at a given point. A point P satisfying F(u,v) = 0 is **singular of multiplicity m** if all derivatives up to order m-1 vanish at P and at least one derivative of order m is non-zero.

A point P of the curve at which both first order derivatives, F_u and F_v, vanish and at least one of the second order derivatives is non-zero is called a **double** point. Similarly, a point P of the curve at which all derivatives up to the second order vanish and at least one of the third order derivatives F_{uuu}, F_{uuv}, F_{uvv}, F_{vvv} is non-zero is called a **triple** point.

A further classification is made within each of the above categories of singularity according to the nature of the solutions of the equation for the tangent line direction(s) at the point of singularity. For a double point (u',v') of the curve, the equation providing the direction cosines α and β of the tangents to the curve at (u',v') is given by [49]

$$\alpha^2 F_{uu} + 2\alpha\beta F_{uv} + \beta^2 F_{vv} = 0 \tag{88}$$

where the second order derivatives are evaluated at (u',v'). Equation (88) is a quadratic equation for α/β or β/α and has two solutions which may be real and distinct, real and coincident, or complex conjugates. Correspondingly, the double point may be classified into the above three categories by the sign of the discriminant, Δ as

$$\Delta = F_{uv}^2 - F_{uu}F_{vv}$$

giving

$\Delta > 0$ *for self–intersection points (or nodes or crunodes)*

$\Delta = 0$ *for cusps (or spinodes)*

$\Delta < 0$ *for isolated points (or hermit points or acnodes)* (89)

The double point is the simplest of singular points. Higher order singularities may be similarly classified by use of the equation describing the tangent line directions at these points in terms of higher order derivatives [18].

IV. Elimination in the Bernstein Basis

In this Appendix, we outline the elimination process of algebraic geometry for two polynomials expressed in the Bernstein basis and determine the relationship between their coefficients so that they have a common root [43].

Let P and Q be two polynomials of degree m and n in the variable u given by

$$P(u) = \sum_{i=0}^{m} a_i' B_{i,m}(u) \tag{90}$$

$$Q(u) = \sum_{i=0}^{n} b_i' B_{i,n}(u) \tag{91}$$

where the coefficients a_i' and b_i' are constants with respect to u. In general, they could be functions of other variables as in the case explained in the body of the report where these coefficients are themselves polynomials of a different variable. The basis functions $B_{i,m}(u)$ and $B_{i,n}(u)$ are Bernstein polynomials of degree m and n respectively. We now rewrite the above polynomials in terms of a different set of basis functions in which the combinatorial factor of the Bernstein basis is missing so that the set of such functions $L_{i,k}(u)$ of degree k is given by

$$L_{i,k}(u) = u^i(1-u)^{k-i} \quad for\ i = 0,1,...k \tag{92}$$

Therefore polynomials (90) and (91) become

$$P(u) = \sum_{i=0}^{m} a_i L_{i,m}(u) \tag{93}$$

$$Q(u) = \sum_{i=0}^{n} b_i L_{i,n}(u) \tag{94}$$

where

$$a_i = a_i' \binom{m}{i} \quad and \quad b_i = b_i' \binom{n}{i} \tag{95}$$

We now form a set of n auxiliary equations by multiplying (93) by the homogenous polynomials $(1-u)^i u^{(n-1)-i}$, $i = 0,1...n-1$ and a set of m auxiliary equations by multiplying (94) by the homogenous polynomials $(1-u)^i u^{(m-1)-i}$, $i = 0,1...m-1$ to get m+n equations of degree m+n-1 in the basis functions (92). Expressing these equations in matrix form we have,

$$\begin{bmatrix} a_0 & a_1 & \cdot & \cdot & \cdot & \cdot & \cdot & a_m & & & & \\ & a_0 & a_1 & \cdot & \cdot & \cdot & \cdot & \cdot & a_m & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & a_0 & a_1 & \cdot & \cdot & \cdot & \cdot & \cdot & & a_m \\ b_0 & b_1 & \cdot & \cdot & \cdot & b_n & & & & & & \\ & b_0 & b_1 & \cdot & \cdot & \cdot & b_n & & & & & \\ & & & \cdot & \cdot & \cdot & \cdot & \cdot & & & & \\ & & & & & b_0 & b_1 & \cdot & \cdot & \cdot & & b_n \end{bmatrix} \begin{bmatrix} (1-u)^{m+n-1}u^0 \\ (1-u)^{m+n-2}u^1 \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ (1-u)^1u^{m+n-2} \\ (1-u)^0u^{m+n-1} \end{bmatrix} = 0 \qquad (96)$$

which can be rewritten compactly as

$$[\,A\,][\,L^{m+n-1}(u)\,] = [\,0\,] \qquad (97)$$

where a_i, $i=0,1,\ldots m$ are involved in first n rows of A and b_i, $i=0,1,\ldots n$ in the last m rows and the blank locations denote zeros. In order that the above set of equations in the variables, which are the elements of the column vector L, have non-trivial solutions, we must have the condition that $|A| = 0$ where $|\;|$ denotes determinant. This is a condition that gives a relation among the coefficients of the original polynomials so that they have a common root. In the particular case, where this condition is used to eliminate one of two variables from two polynomial equations in two variables, the coefficients a_i and b_i are polynomials in the variable that is not being eliminated. This thus leads to a determinant which is a polynomial in one variable and the condition leads to the solution of a univariate polynomial. In general, if $a_i(u)$ is of degree p and $b_i(u)$ is of degree q then the determinant is of degree np+mq [18].

V. Kahan's Summation Technique

In this Appendix, we summarize a technique proposed by Kahan [29] to sum a series of generally unordered numbers in floating point arithmetic which gives a relative error on the sum which is independent of the number of terms, N, in the summation, when $N << 1/\varepsilon$, where ε is the machine precision.

Let us denote the sum S by

$$S = \sum_{i=1}^{N} a_i \tag{98}$$

where the a_i are an unordered array of floating point numbers and N is the number of terms in the sum. Such a summation when performed in floating point arithmetic with a program of the form

```
      S = 0
      do 1000 i = 1,N
        S = S + A(i)
1000  continue
```

can be rewritten symbolically as [29, 14]

$$S = \sum_{i=1}^{N} (1 + \sigma_i)\, a_i \quad where\ |\sigma_i| \leq (1 + \varepsilon)^{N-i} \tag{99}$$

where ε is the relative error incurred in storing a real number ie. the machine precision [50]. For example, in a DEC Vax Station II, the number $\varepsilon = 2.8 \times 10^{-17}$. The relative error incurred in the sum is bounded approximately by $N\varepsilon$ and therefore when the value of N is comparable to the $1/\varepsilon$ then the relative error in the sum is of the same order of magnitude as the sum itself and such summations become unreliable.

Kahan proposed a technique by which such summations could be made without such a quick loss of accuracy in the sum. The key portion of the algorithm consists of obtaining the error committed at each stage and adding it to the summation at the next stage. In floating point addition of two numbers A and B, one of the numbers whose exponent is lower is converted so that it has the matching exponent of the other number. The mantissa of the two numbers are then added and normalized to give the final sum C. In the process of such exponent conversion, a small fraction of the first number is lost (say E) which may be recovered by computing the following floating point sum

$$E = B - (A + B) + A \tag{100}$$

In a continuous sum of a large set of numbers, this error incurred in the kth stage is added to the (k+1)th term to gain some accuracy in the (k+1)th stage. These ideas are incorporated in the following algorithm first published by Kahan [29]

```
            subroutine kahansum(array, nterms, sum)

            double precision array(nterms), sum
            integer nterms

            double precision s,c,y,t
            integer signy,signs,j

            s = 0.0
            c = 0.0

            do 999 j=1,nterms

              y = c + array(j)

              t = s + y

              c = (s - t) + y

999         s = t

            sum = s + c

            return
            end
```

The actual program as published contains some additional statements to account for the differences between various computers. The basic loop involves the addition of the error in the (k-1)-th stage to the k-th term and then performing the addition for the k-th stage. This sum and the components of this sum ie. the k-th term and (k-1)-th partial sum are then used to obtain the error incurred in the kth stage. The relative error incurred in such a summation is only $a\,\varepsilon + O(N\varepsilon^2)$ where a is a small constant. This implies that the relative error in the sum is almost independent of N (ie. linear terms in ε do not contain N).

VI. Oslo Algorithm

In this Appendix, we outline the polyhedron refinement algorithm used in the arbitrary subdivision of B-spline surfaces due to [13, 35].

Let us consider the set of B-spline basis functions $N_{i,k}(u)$ of order k (degree k-1) defined over a knot vector $U = \{u_i, i=0,1...m+k\}$ normally evaluated using the Cox-DeBoor algorithm [15]. We now wish to compute a new set of B-spline basis functions $M_{i,k}(u)$ of the same degree but now defined over a finer partition of the original knot vector $U' = \{u'_i, i=0,1,...p+k\}$. i.e. $U \subseteq U'$ where $\subseteq$ denotes subset. The two basis functions are related by

$$[N] = [M][A] \tag{101}$$

where M is a row vector of the p+k+1 B-spline basis functions of degree k on U' and N is the row vector of the m+k+1 B-spline basis functions of degree k on U. A is a matrix of size (p+k+1)x(m+k+1) and provides the transformation between the two basis functions. Each element of A is called a discrete B-spline denoted by $a_{ij} = \alpha_{jk}(i)$ and may be computed as detailed in [13, 35] using the following recursive definition

$$\begin{aligned} \alpha_{i1}(j) &= 1, \quad u_i \le u'_j < u_{i+1} \\ &= 0, \quad \textit{otherwise} \end{aligned}$$

And for $k \ge 2$

$$\alpha_{ik}(j) = (u'_{j+k-1} - u_i)\beta_{i,k-1}(j) + (u_{i+k} - u'_{j+k-1})\beta_{i+1,k-1}(j)$$

where

$$\begin{aligned} \beta_{ik}(j) &= \frac{\alpha_{ik}(j)}{u_{i+k} - u_i}, \quad u_{i+k} > u_i \\ &= 0, \quad \textit{otherwise} \end{aligned} \tag{102}$$

The general rational B-spline surface of degree k and l in variables u and v may be defined by the following equation

$$\mathbf{Q}_{k,l}(u,v) = [N^u][\mathbf{P}][N^v]^T \tag{103}$$

where $[N^u]$ and $[N^v]$ are row vectors of the m+1 and n+1 basis functions of degree k-1 and l-1 in u and v defined over the knot sets $\{u_i, i=0,1..m+k+1\}$ and $\{v_i, i=0,1...n+l+1\}$, respectively. [P] is a matrix of size (m+1)x(n+1) of control points expressed in homogeneous coordinate form. If a finer partition of the above knot vectors is given then it is possible to obtain a new set of control points that describe the same geometry as that of the surface (103). Let the new knot vectors be denoted by $\{u_i', i=0,1...m'+k+1\}$ and $\{v_i', i=0,1...n'+l+1\}$. The new B-spline basis functions, M^u and M^v, defined over the finer knot vectors in the u and v direction are then given by

$$\begin{aligned}[N^u] &= [M^u][A^u] \\ [N^v] &= [M^v][A^v]\end{aligned} \tag{104}$$

Substituting (104) in (103) we get

$$\mathbf{Q}_{k,l}(u,v) = [M^u][A^u][\mathbf{P}][A^v]^T[M^v]^T \tag{105}$$

The matrix of the new control point coordinates [D] is then obtained by multiplication of three matrices

$$[\mathbf{D}] = [A^u][\mathbf{P}][A^v]^T \tag{106}$$

where $[A^u]$ is matrix of size $(m'+k+1)\mathrm{x}(m+k+1)$ and $[A^v]$ is a matrix of size $(n'+l+1)\mathrm{x}(n+l+1)$. Finally [D] is a matrix of size $(m'\mathrm{x}n')$ where each element of the matrix is a 4D vector of homogeneous coordinate control points.

VII. Root Finding in the Bernstein Basis

In this Appendix, we outline a method of finding roots of univariate polynomials when they are expressed in the Bernstein basis the basic ideas of which was suggested by Lane and Riesenfeld [32].

Let a univariate polynomial P(u) of degree n be given by

$$P(u) = \sum_{i=0}^{n} c_i B_{i,n}(u) \tag{107}$$

where $u \in [0,1]$ and $B_{i,n}(u)$ are the Bernstein polynomials of degree n and the c_i are constant coefficients. We now define the control polygon for the above polynomial to be the ordered sequence of points (x_i,y_i), where $x_i = i/n$ and $y_i = c_i$.

Let the Z be the number of zeros of the above polynomial in [0,1] and V be the number of sign changes in the sequence $\{c_i,\ i = 0,1,...n\}$. Then the following relation holds between the two integer numbers

$$Z = V - 2k \tag{108}$$

where k is a integer between 0 and the integer part of n/2 [20]. This property of the Bernstein basis is known as the variation diminishing property [20] which implies that the polynomial oscillates less than the polygon about the axis $y = 0$. Since Z is a non-negative integer this property can be used to derive the following special cases

$$\begin{array}{l} V = 0 \Rightarrow Z = 0 \\ V = 1 \Rightarrow Z = 1 \end{array} \tag{109}$$

These are used to eliminate intervals containing no roots and to isolate intervals containing only one root.

The coefficients c_i defining the polynomial in [0,1] may be also used to split the curve into two pieces [0,1/2] and [1/2,1] using the Casteljau's algorithm [10, 20]. This algorithm provides the coefficients that individually describe the two pieces of the polynomial.

The algorithm to compute the roots of the polynomial consists of two stages

1. A procedure of isolating intervals so that each interval has only one root or is of such small width that no more subdivision is meaningful
2. Given the interval find the roots by using the intersection of the polygon as an

approximation and progressively narrowing the interval

The first part consists of the following recursive algorithm

```
ISOLATE($c_i$,$u_a$,$u_b$)

Compute the number V for the {$c_i$, i=0,1,...n}

if    ( V = 0 )
    No roots in this interval

else if ( V = 1 )
    There is one  and only one root in this interval, store the interval
    and the number of roots in it and return

else if ( The interval is small enough )
    There is more than one root in this interval, store the interval and
    the number of potential roots in it  and return

else
    Compute $u_m = 0.5(u_a+u_b)$

    Find the two sets of coefficients $c_i^a$ and $c_i^b$
     corresponding to the intervals [$u_a$,$u_m$] and [$u_m$,$u_b$]

    ISOLATE($c_i$,$u_a$,$u_m$)
    ISOLATE($c_i$,$u_m$,$u_b$)
```

This algorithm differs from that presented in [32] in that multiple roots are expected and must be handled in the best manner. The condition that the interval is small enough so as not to warrant further recursion to isolate the roots is based on the idea that it is not worth computing smaller intervals any further if a typical value of the polynomial within the interval is smaller than an estimate of the round-off error incurred in evaluating the polynomial using de Casteljau's algorithm.

The intervals with only one root is then used to find the root in that interval by the following algorithm

```
Procedure FINDROOT($c_i$, $u_c$,$u_d$)

If ( the end coefficients are within a specified tolerance of zero)
Then the corresponding end of the interval is the root

Compute a new u corresponding to the intersection of the polygon with
the axis y = 0.

Find the two sets of coefficients $c_i^a$ and $c_i^b$
```

```
corresponding to the intervals [$u_a$,u] and [u,$u_b$]

Compute the number of sign changes $V_a$ and $V_b$ in each of the two sets of coefficients

if ($V_a$ = 1) Call FINDROOT($c_a$,$u_a$,u)
if ($V_b$ = 1) Call FINDROOT($c_b$,u,$u_b$)
```

A tolerance for floating point computation of these roots as required in the first line of the above algorithm is obtained by an estimate of the error incurred in one evaluation of the polynomial using de Casteljau's algorithm. An upper bound of such an error is given in [20] as

$$2\,n\,c_m\eta \tag{110}$$

where c_m is the maximum of the coefficients in the set $\{c_i\}$ and η is the machine unit of relative error.

References

[1] Abhyankar, S. S., Bajaj, C.
Automatic Parametrization of Rational Curves and Surfaces III: Algebraic Plane Curves.
Report No. CSD-TR-619, Department of Computer Sciences, Purdue University , August, 1986.

[2] Adams, D. A.
A Stopping Criterion for Polynomial Root-Finding.
Communications of the Association for Computing Machinery 10:655-658, 1967.

[3] Arnon, D. S.
Algorithms for the Geometry of Semi-Algebraic Sets.
PhD thesis, University of Wisconsin - Madison, August, 1981.

[4] Arnon, D. S.
Topologically Reliable Display of Algebraic Curves.
ACM Computer Graphics 17(3):219-227, July, 1983.

[5] Arnon, D. S., Collins, G. E., McCallum, S.
Cylindrical Algebraic Decomposition I: The Basic Algorithm.
SIAM Journal of Computation 13(4):865-877, November, 1984.

[6] Arnon, D. S., Collins, G. E., McCallum, S.
Cylindrical Algebraic Decomposition II: An Adjacency Algorithm for the Plane.
SIAM Journal of Computation 13(4):878-889, November, 1984.

[7] Bajaj, C., Hoffmann, C. M., Hopcroft, J.
Tracing Planar Algebraic Curves.
Report No. CSD-TR-637, Department of Computer Sciences, Purdue University , September, 1987.

[8] Bajaj, C., Dyksen, W. R., Hoffmann, C. M., Houstis, E. N., Korb, J. T., Rice, J. R.
Computing about Physical Objects.
Report No. CSD-TR-696, Department of Computer Sciences, Purdue University , July, 1987.

[9] Barnhill, R. E., Farin, G., Jordan, M., Piper, B. R.
Surface/Surface Intersection.
Computer Aided Geometric Design 4:3-16, 1987.

[10] Boehm, W., Farin, G., Kahmann, J.
A Survey of Curve and Surface Methods in CAGD.
Computer Aided Geometric Design 1(1):1-60, July, 1984.

[11] Casale, M. S.
Free-Form Solid Modeling with Trimmed Surfaces Patches.
IEEE Computer Graphics and Applications 7(1):33-43, January, 1987.

[12] Chryssostomidis, C., Patrikalakis, N. M.
Geometric Modeling in Computer Aided Design of Marine Systems.
Proceedings of the Third International Symposium on Practical Design of Ships and Mobile Units at The Norwegian Institute of Technology, Trondheim, Norway , June, 1987.

[13] Cohen, E., Lyche, T., Riesenfeld, R.
Discrete B-Splines and Subdivision Techniques in Computer-Aided Geometric Design and Computer Graphics.
Computer Graphics and Image Processing 14:87-111, 1980.

[14] Dahlquist, G. and Bjorck, A.
Numerical Methods.
Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.

[15] De Boor, C.
On Calculating with B-Splines.
Journal of Approximation Theory 6:50-62, 1972.

[16] Farouki, R. T., Hinds, J. K.
A Hierarchy of Geometric Forms.
IEEE Computer Graphics and Applications 5(5):51-78, May, 1985.

[17] Farouki, R. T.
Trimmed Surface Algorithms for the Evaluation and Interrogation of Solid Boundary Representations.
IBM Research Report, RC 12052 , August, 1986.

[18] Farouki, R. T.
The Characterization of Parametric Surface Sections.
Computer Vision, Graphics and Image Processing 33:209-236, 1986.

[19] Farouki, R. T.
Computational Issues in Solid Boundary Evaluation.
IBM Research Report RC 12454 (#55990) , January, 1987.

[20] Farouki, R. T., Rajan, V. T.
On the Numerical Condition of Bernstein Polynomials.
IBM Research Report RC 12626 (#56733) , March, 1987.

[21] Geisow, A.
Surface Interrogations.
PhD thesis, School of Computing Studies and Accountancy, University of East Anglia, Norwich NR47TJ, U. K., July, 1983.

[22] Gordon, W. J., Riesenfeld, R. F.
B-Spline Curves and Surfaces.
Computer Aided Geometric Design, Edited by Barnhill, R., and Riesenfeld, R. F., Academic Press, Inc., 1974.

[23] Grant, J. A., Hitchins, G. D.
An Always Convergent Minimization Technique for the Solution of Polynomial Equations.
Journal of Industrial and Mathematical Applications 8:122-129, 1971.

[24] Grossman, D. D.
Computer Applications in Manufacturing.
IBM Research Report RC 11662 , January, 1986.

[25] Hanrahan, P.
Ray Tracing Algebraic Surfaces.
ACM Computer Graphics 17(3):83-90, July, 1983.

[26] Hildebrand, F. B.
Advanced Calculus for Applications.
Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[27] Hoffmann, C. M.
Algebraic Curves.
Report No. CSD-TR-675, Department of Computer Sciences, Purdue University , May, 1987.

[28] Jenkins, M. A.
Algorithm 493.
Transactions on Mathematical Software 1:178, 1975.

[29] Kahan, W.
A Survey of Error Analysis.
Proceedings of the International Federation for Information Processing Congress 1971 2:1214-1239, August, 1971.

[30] Knuth, D. E.
The Art of Computer Programming, Vol. 2, Seminumerical Algorithms.
Addison-Wesley, Reading, Massachusetts, 2nd Edition, 1981.

[31] Lane, J. M., Riesenfeld, R. F.
A Theoretical Development for the Computer Display and Generation of Piecewise Polynomial Surfaces.
IEEE Transactions PAMI 2 , 1980.

[32] Lane, J. M., Riesenfeld, R. F.
Bounds on polynomials.
IEEE Computer Graphics and Applications , 1981.

[33] Lawrence, J. D.
A Catalogue of Special Plane Curves.
Dover Publications, Inc., New York, 1972.

[34] Levin, J. Z.
Mathematical Models for Determining the Intersections of Quadric Surfaces.
Computer Vision, Graphics and Image Processing 11:73-87, 1979.

[35] Lyche, T., Cohen, E., Morken, K.
Knot Line Refinement Algorithms for Tensor Product B-Spline Surfaces.
Computer Aided Geometric Design 2:133-139, 1985.

[36] Symbolics Inc.
VAX UNIX MACSYMA Reference Manual.
Symbolics Inc., 1985.

[37] NAG.
Numerical Algorithms Group FORTRAN Library.
NAG, Oxford, England, 1985.

[38] Patrikalakis, N. M.
Algebraic Curves in Geometric Modeling Applications.
Submitted for Publication , February, 1987.

[39] Petersen, C. S.
Adaptive Contouring of Three-Dimensional Surfaces.
Computer Aided Geometric Design 1:61-74, January, 1984.

[40] Sederberg, T. W.
Planar Piecewise Algebraic Curves.
Computer Aided Geometric Design 1:241-255, 1984.

[41] Sederberg, T. W., Anderson, D. C., Goldman, R. N.
Implicit Representation of Parametric Curves and Surfaces.
Computer Vision, Graphics and Image Processing 28(1):72-84, 1984.

[42] Sederberg, T. W.
Piecewise Algebraic Surface Patches.
Computer Aided Geometric Design 2:53-59, 1985.

[43] Sederberg, T. W., Parry, S. R.
Comparison of Three Curve Intersection Algorithms.
Computer Aided Design 18(1):58-63, January, 1986.

[44] Solomon, B. J.
Surface Intersection for Solid Modelling.
PhD thesis, Clare College, University of Cambridge, Cambridge, England, 1985.

[45] Sommerville, D. M. Y.
Analytical Geometry of Three Dimensions.
Cambridge University Press, Cambridge, 1951.

[46] Thomas, S. W.
Modeling Volumes Bounded by B-Spline Surfaces.
PhD thesis, Department of Computer Science, University of Utah, Salt Lake City, Utah, June, 1984.

[47] Tiller, W.
Rational B-Splines for Curve and Surface Representation.
IEEE Computer Graphics and Applications 3(6):61-69, September, 1983.

[48] Varady, T.
Surface-Surface Intersections for Double-Quadratic Parametric Patches in a Solid Modeller .
Proceedings of the U.K. - Hungarian Seminar on Computational Geometry for CAD/CAM, Cambridge University , 1983.

[49] Walker, R. J.
Algebraic Curves.
Princeton University Press, Princeton, New Jersey, 1950.

[50] Wilkinson, J. H.
Rounding Errors in Algebraic Processes.
H. M. Stationery Office, London, England, 1963.

[51] Wilson P. R.
Solid Modeling R & D in the USA.
European Conference on Solid Modeling, London , September, 1985.